

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 7, 2011

P. Hoffman
VPN Consortium
J. Schlyter
Kirei AB
W. Kumari
A. Langley
Google
October 4, 2010

Using Secure DNS to Associate Certificates with Domain Names For TLS
draft-hoffman-keys-linkage-from-dns-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 certificate 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

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 April 7, 2011.

Copyright Notice

Copyright (c) 2010 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

Internet-Draft

DNS Cert Linkage for TLS

October 2010

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.

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 the trust anchor upon which the server's certificate is rooted, and must perform correct validation on the certificate.

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".

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.

In this document, a certificate association is based on a cryptographic hash of a certificate (sometimes called a "fingerprint"). That is, a hash is taken of the certificate, and that hash is the certificate association. The type of hash function used can be chosen by the DNS administrator. (Note that there may be other methods to securely obtain certificate associations in DNS, but those methods are not covered by this document.)

Certificate associations are made between a 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.

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.

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.

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

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 is being discussed on the "keyassure" mailing list; see [<https://www.ietf.org/mailman/listinfo/keyassure>](https://www.ietf.org/mailman/listinfo/keyassure).

[2.](#) Getting TLS Certificate Associations from the DNS with the CERT RR

The CERT RR [[RFC4398](#)] allows expansion by defining new certificate types. This document describes two new Certificate Types, TLSFP and TLSRQ. A query on a domain name for the CERT RR can return one or more records of the type CERT, and zero or more of those CERT responses can be of type TLSFP and TLSRQ.

[2.1.](#) The TLSFP Certificate Type

This section describes the TLSFP certificate type of the CERT RR. The TLSFP certificate type is TBD1. The key tag and algorithm fields are both set to zero.

The format of the TLSFP certificate is a binary record, which MUST be in the order defined here, is:

- o A one-octet value, called "hash type", specifying the type of hash algorithm used for the certificate association. This value has the same values as those of the TLS hash, as defined in the IANA registry titled "TLS HashAlgorithm Registry" (<http://www.iana.org/assignments/tls-parameters>). For example, the value for the SHA-1 hash function is "2".
- o A one-octet value, called "certificate type", specifying the TLS certificate type of the target certificate. This value has the same values as those of the TLS certificate types, as defined in the IANA registry titled "TLS Certificate Types" (<http://www.iana.org/assignments/tls-extensiontype-values>). For example, the value for PKIX certificates is "0".
- o A variable-length set of bytes containing the hash of the associated certificate (that is, of the certificate itself, not the TLS ASN.1Cert object).

An example of a fingerprint for a single certificate:

```
www.example.com. IN CERT
    TLSFP 0 0 AgDne3GdTpxjwLCgMzvgpBi0SQthjg==
```

An example of a fingerprint of a certificate that is found by its hash value:

e77b719d4e9c63c0b0a0333be0a4188e490b618e.www.example.com. IN CERT
TLSFP 0 0 AgDne3GdTpxjwLCgMzvgpBi0SQthjg==

A note on terminology: Some people have said that TLSFP is a form of "certificate exclusion". This is true, but in a very unusual sense. That is, a DNS reply that contains one TLSFP certificate type inherently excludes every other possible certificate in the universe other than those found with a pre-image attack. The TLSFP certificate type is better thought of as "enumeration" of a small number of certificate associations, not "exclusion" of a near-infinite number of other certificates.

[2.2.](#) The TLSRQ Certificate Type

This section describes the TLSRQ certificate type of the CERT RR. If a domain has many certificates associated with it, the number of TLSFP CERT RRs may become impractical. In this case, a TLSRQ certificate type may be used.

The semantics of the TLSRQ certificate type are that the requesting party should send another query for the CERT RR that is formed by prepending a label to the host name that is the hex of the SHA-1 hash value taken over the certificate (not the TLS ASN.1Cert object). If that certificate is a valid certificate that would have been returned in a TLSFP certificate type, requesting it with this encoded hash prepended to the host name will yield a TLSFP certificate type. There is no security problem with using SHA-1 even if the SHA-1 hash function continues to weaken because the hash is simply used to differentiate the various certificates used by the server.

Note that the TLSRQ certificate type can only be used if the requesting party knows the hash of the certificate that is being used by the TLS server. This usually means that the request will be sent by the application acting as the TLS client after it has received the TLS server's Certificate message that contains the server's certificate. Such a request can slow down the TLS handshake processing, but is required in the case where different hosts with different certificates respond on the same domain name.

The typical scenario would look like the following:

1. The application client that is about to use TLS sends a CERT query for `www.example.com`.
2. The name server responds with a CERT record that has a TLSRQ certificate type.
3. The application client starts the TLS handshake, and receives a Certificate message from the server.
4. The application client takes the SHA-1 hash of the certificate, encodes that value as hex. For this example, assume that encoded value is `"e77b719d4e9c63c0b0a0333be0a4188e490b618e"`.
5. The application client sends a CERT query for `e77b719d4e9c63c0b0a0333be0a4188e490b618e.www.example.com`. It receives a response with a TLSFP certificate type.
6. The application uses the information in the TLSFP certificate type and associates it with `www.example.com`.

The TLSRQ certificate type has a certificate body with a single octet of 0. The TLSFP certificate type is TBD2.

For example:

```
www.example.com. IN CERT TLSRQ 0 0 AA=
```

[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.

If a certificate association contains a hash type that is not understood by the TLS client, that certificate association **MUST** be completely ignored.

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 process 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.

[3.1.](#) Certificate Validation by TLS Clients When Using Certificate Associations

TLS client policy is deliberately not prescribed by this specification. A client **MAY** choose to trust a DNSSEC-secured certificate association, depending on its local policy.

[3.1.1.](#) Use of Self-Signed Certificates

One expected use case for this protocol is that some TLS servers will begin to use self-signed certificates in association with certificate associations. A TLS client that is using this protocol needs to treat self-signed certificates as special, and thus **SHOULD NOT** attempt certificate validation on them. (An exception to this rule would be clients that keep self-signed end entity certificates in its trust anchor store.)

[3.1.2.](#) Ignoring Host Names in Certificates

All data in a self-signed certificate other than the key itself can be ignored as untrusted unless a client validates the self-signed certificate to a trust anchor that is identical to the certificate. That means that the host name given in the self-signed certificate is meaningless, and that the only way to associate the public key in the certificate with the domain name is through the certificate association made in the DNS, secured with DNSSEC.

If a TLS client fully trusts the association between a domain name and the certificate that was provided by the DNS, then that client **MUST** ignore the domain name that is given in the certificate. That is, the certificate might contain a domain name that is different than the one that was used to get the TLSFP record, but if the client is trusting the TLSFP record, it doesn't matter what domain name is used in the certificate. An expected use case for this protocol is to allow someone who controls the private key on a certificate to use

that certificate for multiple TLS servers. These servers might be on a single computer that has many domain names (such as a computer that is both a web host and a mail host, and is known by both "www.example.com" and "smtp.example.com"), or they might be on different computers (such as multiple computers that all respond IP addresses reachable as "www.example.com").

3.1.3. Use of Local Trust Anchors

Another expected use case for this protocol is that some TLS servers will use certificates that chain to a trust anchor that might not be one that is trusted by the TLS client, such as a local certificate authority (CA) that is administered by the organization that runs the TLS server. Because of this, a TLS client that is using this protocol that performs certificate validation on server certificates MAY have a method to communicate with the user that differentiates between validation failures that occur on certificates that have had secure certificate associations and those that have not. If it does not have such a method of communication, the failure to validate SHOULD cause the same error as for any other certificate validation.

3.1.4. Use of Additional Certificate Data

Some TLS clients extract data from the certificate other than the key to show to the user; for example, most modern web browsers have the ability to show an extended validation (EV) name that is embedded in a certificate. Because this data comes from a trusted third party and not the TLS server itself, TLS clients that extract additional information from TLS server certificates MUST validate those certificates in the normal fashion.

4. IANA Considerations

This document requests that IANA allocates two certificate types from the CERT RR certificate type registry (<<http://www.iana.org/assignments/cert-rr-types>>). The types are to be allocated from the 'IETF Consensus' range.

Type: TLSFP
Meaning: TLS certificate associations

Decimal type: TBD2
Type: TLSRQ
Meaning: Client should request TLS certificate associations
retrieved using hashes

[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 CERT records.

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

The SHA-1 hash used in the queries after the TLSRQ certificate type is only used to differentiate certificates. If there is a collision between the SHA-1 hashes of two certificates used by the servers that are at the host name, there is no problem because both of those certificates will have the same association to the domain name.

The values of the TLSFP and TLSRQ records 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 TLSFP and TLSRQ records are 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 TLSFP and TLSRQ records in a zone is weaker than the authentication mechanism for changing the A/AAAA records, an 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.

[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, Simon Josefsson, Phill Hallam-Baker, Ilari Liusvaara, among others.

[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.
- [RFC4398] Josefsson, S., "Storing Certificates in the Domain Name System (DNS)", [RFC 4398](#), March 2006.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), August 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.

[Appendix A](#). Ideas Considered But Not Necessarily Chosen

This appendix will list some of the ideas that have been kicked around in this space and give a few paragraphs why they weren't chosen for the current version this proposal. The following is a placeholder for the list that will be filled out more in future versions of this document:

- o A flag that indicates that the certificate with the associated key must be signed by a trusted CA. Briefly: this was not added because it is up to the TLS server to decide which type of certificate it wants to serve up. Serving a self-signed certificate would effectively disable traditional certificate validation, whereas serving a certificate signed by a trusted CA would require both validation by DNSSEC and the trusted CA.
- o A flag that indicates that all connections to this server need to be TLS secured. Briefly: this is a good idea but it is not related to the key of the certificate given in TLS, and thus should be indicated in a different method.
- o Giving keys instead of hashes of keys. Briefly: TLS requires that the server gives a certificate, and some systems use the metadata from a CA-signed certificate for display, so there is value to always looking in the certificate.
- o Hashes of keys vs. hashes of certificates. Briefly: we have changed our minds (at least once) on this. Our original thinking was that there are many reasons why someone might change their certificate while leaving the public key alone, and those changes should not have to force them to change the DNS record because they do not actually change what the TLS client cares about; thus, use hashes of keys. Our new thinking is that there are certificate semantics that we want to pass (namely, should the client actually do the certificate validation), and attaching those semantics to keys is confusing; thus, use hashes of certificates.
- o List TLS/DTLS ports or services for which the certificate is

associated. Briefly: we had this in an earlier version of this document but got rid of it when it was pointed out that this is an edge case, and most servers differentiate these services by domain names such as "mail.example.com" and "www.example.com".

- o Different ways of encoding this information in the DNS. Briefly: we considered a new RR type and coming up with an encoding of the TXT RR type, but didn't see any significant advantage of them over using the CERT RR, and there were disadvantages. A disadvantage

of a new RR type is getting DNS servers and clients to recognize it; a disadvantage of coming up with a new TXT format is that doing so prevents wildcards. There is a lot more to discuss here, but the authors are now happy with a new sub-type for the CERT RR.

- o Having the hash be over the TLS certificate structure instead of just the end-entity certificate. Briefly: the TLS certificate structure currently allows a chain of PKIX certificates, and the semantics of what is being associated in a chain is not clear. Further, the structure might be changed in the future (such as to allow a group of web-of-trust OpenPGP certificates), and the semantics of what is being associated would become even less clear.
- o Having an "always uses TLS" flag in the TLSFP record. Briefly: the policy of always using TLS should be carried elsewhere because it does not line up exactly with TLSFP. Having this flag in the TLSFP record can lead to silly states if a site has multiple TLSFP records that have the flag set differently. It is completely unclear what such a flag will mean for SMTP, which uses a STARTTLS mechanism built into the unprotected protocol. If the TLSFP record does not apply to a specific service, then all services on that host that could use TLS must do so or not do so together. Note, however, that we believe that an "always uses TLS" statement should be available in the DNS.

[Appendix B](#). Changes between -00 and -01

Change the association from being a hash of the key of a PKIX certificate to being a hash of a certificate (PKIX or other). This, of course, makes large changes throughout the document.

Expanded the document to cover DTLS as well.

Added a pointer to the keyassure mailing list.

Removed the proposals for two alternate formats (the TLSFP Resource Record and the TXT record encoding). Added a bit to [Appendix A](#) about this.

Got rid of the specification for ports within a single domain name.

Made the hash type one octet and used the DS registry instead of defining our own.

Added "Necessarily" chosen in the title of [Appendix A](#) to show that we might (continue to) change our minds after discussion.

Hoffman, et al.

Expires April 7, 2011

[Page 11]

Internet-Draft

DNS Cert Linkage for TLS

October 2010

Added Simon Josefsson to the acknowledgements.

[Appendix C](#). Changes between -01 and -02

Added the TLSRQ certificate type and its semantics.

Pointed to the IANA registry for DS hash types.

Added Phill Hallam-Baker to the acknowledgements.

[Appendix D](#). Changes between -02 and -03

In 1, added "In this document" to clarify that there may be other types of certificate associations described elsewhere. Also moved the sentence from 3 to 1 to help make that point earlier.

Added a paragraph to 1 to emphasize that multiple TLSFP records can be returned for cases where different server software on one host uses different certificates.

In 2.1, got rid of "validation preference".

In 2.1, changed the values for the hash type from being from the IANA

registry for DNS to the IANA registry for TLS. This caused a change in the examples from "0" to "2".

In 2.1, added a one-octet value for "certificate type".

At the end of 2.1, added a paragraph about the term "exclusion".

Clarified [section 3](#) to make it clear that the certificate associations being described are only the ones from this document. Also clarified the semantics of finding a match.

Removed the last paragraph of 3 (about validation preference) and created 3.1 to talk about validation.

In 5, added "unless the client performs certificate validation and rejects the certificate" to the second paragraph.

In 5, added a new paragraph about monitoring zone changes visually, and added a new paragraph about separate authentication mechanisms for the TLS-specific records.

Added Ilari Liusvaara to acknowledgments.

In [Appendix A](#), added a paragraph about why we do not have a "always uses TLS" flag in this protocol.

Authors' Addresses

Paul Hoffman
VPN Consortium

Email: paul.hoffman@vpnc.org

Jakob Schlyter
Kirei AB

Email: jakob@kirei.se

Warren Kumari
Google

Email: warren@kumari.net

Adam Langley
Google

Email: agl@google.com