Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 14, 2011

P. Hoffman
VPN Consortium
J. Schlyter
Kirei AB
W. Kumari
A. Langley
Google

August 13, 2010

Using Secure DNS to Associate Keys with Domain Names For TLS
draft-hoffman-keys-linkage-from-dns-00

Abstract

TLS uses PKIX certificates for authenticating the server. Users want their applications to verify that the key in 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 key that appears in a 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 February 14, 2011.

Copyright Notice

1. Introduction

The first response from the server in TLS [RFC5246] may contain a PKIX certificate. In order for the TLS client to authenticate that it is talking to the expected TLS server, the client must validate that the key in 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 one of many places in the PKIX certificate, must trust the trust anchor upon which the server's PKIX certificate is rooted, and must perform correct PKIX validation on the certificate.

Some people want a different way to authenticate the association of the key in 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 public key that might be used by a host at that domain name. The easiest way to do this is to use the DNS.

A key association is a cryptographic hash of the public key in a PKIX certificate (sometimes called a "fingerprint"). That is, a hash is taken of the DER-encoded subjectPublicKeyInfo field of the PKIX certificate as defined in [RFC5280], and that hash is the key association. The type of hash function used can be chosen by the DNS administrator.

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 key in the PKIX certificate that is obtained from the TLS server with a domain name using DNS protected by DNSSEC. Because the key association was retrieved based on a DNS query, the domain name in the query is by

definition associated with the key.

This document only relates to securely getting the DNS information for the key 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 getting keys for TLS; 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].

2. Getting TLS Key Associations from the DNS

This section describes three equivalent methods for encoding TLS associations: a new certificate type of the existing CERT RR (defined in [RFC4398]), a new resource record (RR) called "TLSFP" and a TXT RR that can be emitted when the query has "_tlsfp" as the leftmost label.

EXTREMELY IMPORTANT NOTE: Only one of these methods describe in this document should be selected for the final protocol. We have listed them in our approximate order of preference, but look forward to discussion. When that decision is made, the two methods not used will be moved to the appendix.

2.1. The TLSFP Certificate Type of the CERT RR

The CERT RR [RFC4398] allows expansion by defining new certificate types. The new TLSFP certificate type is defined here. 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.

The format of the TLSFP certificate type is binary. In the record, all integers consist of two bytes in network byte order. The record, which MUST be in the order defined here, is:

- An integer specifying how many port numbers are listed. If this value is zero (0), the key association is valid for any port.

- An optional unordered set of two-byte integers, ranging from 1 to 65535, specifying the TCP/UDP ports for which the key association

is valid.

- An integer specifying the type of hash algorithm used for the key association.

- A variable-length set of bytes containing the hash of the associated key.

For example:

```
www.example.com. IN CERT TLSFP 0 0 ( AQG7ASCWCnpVpwaT
wRsZLt3FmDw45y/8H/Ie9tyEWLd2nZF9 )
```

Note that, unlike the following two format proposals, no version number is needed for the certificate type because a request for a CERT RR can yield multiple results. If there is a later improvement to the TLSFP certificate type, it could be sent along with a TLSFP certificate type in a response.

2.2. The TLSFP Resource Record

The new RR TLSFP resource record is defined here. A query on a domain name for the TLSFP type can return one or more records of the type TLSFP.

The format of the TLSFP response is binary. In the record, all integers consist of two bytes in network byte order. The record, which MUST be in the order defined here, is:

- The version number. This is useful if non-critical changes are made to this RR later. The initial version number is 42.

- An integer specifying how many port numbers are listed. If this value is zero (0), the key association is valid for any port.

- An optional unordered set of two-byte integers, ranging from 1 to 65535, specifying the TCP/UDP ports for which the key association is valid.

- An integer specifying the type of hash algorithm used for the key association.

- A variable-length set of bytes containing the hash of the associated key.

For example:

```
www.example.com. IN TLSFP 42 1 443 1 20960a7a55a706
```



```
93c11b192eddc5983c38e72ffc1ff21ef6dc8458b7769d917d
```

[[This will need a proper RRTYPE definition. That will be added later if this option is chosen.]]

2.3. Using a TXT Resource Record with a _TLSFP Label Prefix

A request for a TXT RR whose domain is the label _tlsfp prepended to a domain name can be used to get the KEY associated with the domain name. A query of this can return one or more records of the type TXT.

The format of the TXT response is ASCII text. The record, which MUST be in the order defined here, is:

- One instance of "ver=", the version number, followed by ";", followed by ";". This is useful if non-critical changes are made to this RR later. The initial version number is 42.

- Zero or more instances of "port=" followed by an TCP/UDP port for which the key association is valid (expressed as an integer), followed by ";". If a port is not specified, the key association

is valid for all ports.

- The type of hash algorithm used for key association, specified as "type=nn;" where "nn" is an integer defined below.

- "hash=" followed by the set of bytes containing the hash of the associated key; the bytes are encoded as lower-case hexadecimal.

For example:

```
_tlsfp.www.example.com. IN TXT "ver=42; port=443; type=1;
hash=20960a7a55a70693c11b192eddc5983c38e72ffc1ff21ef6dc84
58b7769d917d
```

2.4. Key Association Hash Algorithms

The initial list of key association hash algorithms is:

- 0 - reserved

- 1 - SHA2-256 [RFC4634]

- 2 - SHA2-384 [RFC4634]

- 3 - SHA2-512 [RFC4634]

Defining other key association hash types requires IETF consensus as defined in [RFC2434].

For interoperability reasons, as few hash algorithm as possible should be reserved. The only reason to reserve additional types is to increase security.

3. Use of TLS Key Associations from the DNS in TLS

In order to use one or more TLS key associations obtained from the DNS, an application MUST assure that the keys were obtained using DNS protected by DNSSEC. There may be other methods to securely obtain keys in DNS, but those methods are not covered by this document.

An application that requests TLS keys using the method described in the previous section obtains zero or more key associations. If the application receives zero key associations, it process TLS in the normal fashion. If one or more key associations are received from the DNS:

- If the PKIX certificate given by the TLS server is signed by a CA trusted by the client, the application compares each key association with the the hash of the key from the certificate, using the same hash function that is given in the key association type. If a match is found, the TLS handshake continues as normal, including the TLS client doing all PKIX validation checks.

- If the PKIX certificate given by the TLS server is not signed by a CA trusted by the client, the application compares each key association with the the hash of the key from the certificate, using the same hash function that is given in the key association type. If a match is found, the TLS handshake continues using the key from the certificate, but with no PKIX validations checks being performed.

In either of the above cases, if a match between the key association(s) is not found, the TLS client MUST abort the handshake with an "access_denied" error.

4. IANA Considerations

[[TBD. Will include the registration for the TLSFP RR if that is the style chosen, as well as a new registry for hash algorithm types, depending on what style is decided on.]]

5. Security Considerations

[[TBD. This section will need to describe, at least, the "attack" where a DNS administrator goes rogue and changes both the A and TLSFP records for a domain name. Also will discuss the need for secure DNS.]]

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, and Jeff Hodges, among others.

7. References

7.1. Normative References

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

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[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.

[RFC4634] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006.

[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.

Appendix A. Ideas Considered But Not 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 this proposal. The following is a placeholder for the list that will be filled out more in future versions of this document:

- 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 PKIX validation, whereas serving a certificate signed by a trusted CA would require both validation by DNSSEC and the trusted CA.

- 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.

- Giving keys instead of fingerprints. Briefly: TLS requires that the server gives a PKIX certificate, and some systems use the metadata from a CA-signed certificate for display, so there is value to always looking in the certificate.

- After a format for the information is chosen, the other two listed earlier will go into this appendix.

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