TLS M. Shore
Internet-Draft No Mountain Software
Intended status: Standards Track R. Barnes
Expires: December 6, 2016 Mozilla
S. Huque
Verisign Labs
W. Toorop
NLNet Labs
June 4, 2016

A DANE Record and DNSSEC Authentication Chain Extension for TLS
draft-ietf-tls-dnssec-chain-extension-00

Abstract

This draft describes a new TLS extension for transport of a DNS record set serialized with the DNSSEC signatures needed to authenticate that record set. The intent of this proposal is to allow TLS clients to perform DANE authentication of a TLS server certificate without needing to perform additional DNS record lookups. It will typically not be used for general DNSSEC validation of TLS endpoint names.

Table of Contents

1. Requirements Notation

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 [RFC2119].

2. Introduction

This draft describes a new TLS [RFC5246] extension for transport of a DNS record set serialized with the DNSSEC signatures [RFC4034] needed to authenticate that record set. The intent of this proposal is to allow TLS clients to perform DANE authentication [RFC6698] of a TLS server certificate without performing perform additional DNS record lookups and incurring the associated latency penalty. It also provides the ability to avoid potential problems with TLS clients being unable to look up DANE records because of an interfering or broken middlebox on the path between the endpoint and a DNS server. And lastly, it allows a TLS client to validate DANE records itself without needing access to a validating DNS resolver to which it has a

secure connection. It will typically not be used for general DNSSEC validation of endpoint names, but is more appropriate for validation of DANE TLSA records.

This mechanism is useful for TLS applications that need to address the problems described above, typically web browsers or VoIP and XMPP services. It may not be relevant for many other applications. For example, SMTP MTAs are usually located in data centers, may tolerate extra DNS lookup latency, are on servers where it is easier to provision a validating resolver, or are less likely to experience traffic interference from misconfigured middleboxes. Furthermore, SMTP MTAs usually employ Opportunistic Security [RFC7435], in which the presence of the DNS TLSA records is used to determine whether to enforce an authenticated TLS connection. Hence DANE authentication of SMTP MTAs [RFC7672] will typically not use this mechanism.

The extension described here allows a TLS client to request in the client hello message that the DNS authentication chain be returned in the (extended) server hello message. If the server is configured for DANE authentication, then it performs the appropriate DNS queries, builds the authentication chain, and returns it to the client. The server will usually use a previously cached authentication chain, but it will need to rebuild it periodically as described in Section 5. The client then authenticates the chain using a pre-configured trust anchor.

This specification is based on Adam Langley's original proposal for serializing DNSSEC authentication chains and delivering them in an X.509 certificate extension [AGL]. It modifies the approach by using wire format DNS records in the serialized data (assuming that the data will be prepared and consumed by a DNS-specific library), and by using a TLS extension to deliver the data.

3. DNSSEC Authentication Chain Extension

3.1. Protocol

A client MAY include an extension of type "dnssec_chain" in the (extended) ClientHello. The "extension_data" field of this extension MUST be empty.

Servers receiving a "dnssec_chain" extension in the client hello, and which are capable of being authenticated via DANE, SHOULD return a serialized authentication chain in the extended ServerHello message, using the format described below. If a server is unable to return a authentication chain, or does not wish to return a authentication chain, it does not include a dnssec_chain extension. As with all TLS

extensions, if the server does not support this extension it will not return any authentication chain.

3.2. DNSSEC Authentication Chain Data

The "extension_data" field of the "dnssec_chain" extension MUST contain a DNSSEC Authentication Chain encoded in the following form:

```
          opaque AuthenticationChain<0..2^16-1>;
```

The AuthenticationChain structure is composed of a sequence of uncompressed wire format DNS resource record sets (RRset) and corresponding signatures (RRsig) records. The record sets and signatures are presented in validation order, starting at the target DANE record, followed by the DNSKEY and DS record sets for each intervening DNS zone up to a trust anchor chosen by the server, typically the DNS root.

This sequence of native DNS wire format records enables easier generation of the data structure on the server and easier verification of the data on client by means of existing DNS library functions. However this document describes the data structure in sufficient detail that implementers if they desire can write their own code to do this.

[TODO: mention that to reduce the size of the chain, the server can deliver exactly one RRsig per RRset, namely the one used to validate the chain as it is built.]

Each RRset in the chain is composed of a sequence of wire format DNS resource records. The format of the resource record is described in RFC 1035 [RFC1035], Section 3.2.1. The resource records SHOULD be presented in the canonical form and ordering as described in RFC 4034 [RFC4034].

```
          RR(i) = owner | type | class | TTL | RDATA length | RDATA
```

RRs within the RRset are ordered canonically, by treating the RDATA portion of each RR as a left-justified unsigned octet sequence in which the absence of an octet sorts before a zero octet.

The RRsig record is in DNS wire format as described in RFC 4034 [RFC4034], Section 3.1. The signature portion of the RDATA, as described in the same section, is the following:

```
signature = sign(RRSIG_RDATA | RR(1) | RR(2)... )
```

where, RRSIG_RDATA is the wire format of the RRSIG RDATA fields with the Signer's Name field in canonical form and the signature field excluded.

The first RRset in the chain MUST contain the DANE records being presented. The subsequent RRsets MUST be a sequence of DNSKEY and DS RRsets, starting with a DNSKEY RRset. Each RRset MUST authenticate the preceding RRset:

- A DNSKEY RRset must include the DNSKEY RR containing the public key used to verify the previous RRset.

- For a DS RRset, the set of key hashes MUST overlap with the preceding set of DNSKEY records.

In addition, a DNSKEY RRset followed by a DS RRset MUST be self-signed, in the sense that its RRSIG MUST verify under one of the keys in the DNSKEY RRSET.

The final DNSKEY RRset in the authentication chain, containing the trust anchor may be omitted. If omitted, the client MUST verify that the key tag and owner name in the final RRSIG record correspond to a trust anchor. There may however be reason to include the trust anchor RRset and signature if clients are expected to use RFC5011 compliant key rollover functions inband via the chain data. In that case, they will need to periodically inspect flags (revocation and secure entry point flags) on the trust anchor DNSKEY RRset.

For example, for an HTTPS server at www.example.com, where there are zone cuts at "com." and "example.com.", the AuthenticationChain structure would comprise the following RRsets and signatures (the data field of the records are omitted here for brevity):

```
_443._tcp.www.example.com. TLSA
RRSIG(_443._tcp.www.example.com. TLSA)
example.com. DNSKEY
RRSIG(example.com. DNSKEY)
example.com. DS
RRSIG(example.com. DS)
com. DNSKEY
RRSIG(com. DNSKEY)
com. DS
RRSIG(com. DS)
. DNSKEY
RRSIG(. DNSKEY)
```

Names that are aliased via CNAME and/or DNAME records may involve multiple branches of the DNS tree. In this case the authentication chain structure will be composed of a sequence of these multiple intersecting branches. DNAME chains should omit unsigned CNAME records that may have been synthesized in the response from a DNS resolver. Wildcard DANE records will need to include the wildcard name as well as a negative proof (i.e. NSEC or NSEC3 records) that no closer name exists.

A CNAME example:

```
_443._tcp.www.example.com.   IN   CNAME    ca.example.net.
ca.example.net.              IN   TLSA     2 0 1 ...
```

Here the authentication chain structure is composed of two consecutive chains, one for _443._tcp.www.example.com/CNAME and one for ca.example.net/TLSA. The second chain can omit the record sets at the end that overlap with the first.

TLS DNSSEC chain components:

```
_443._tcp.www.example.com. CNAME
RRSIG(_443._tcp.www.example.com. CNAME)
example.com. DNSKEY
RRSIG(example.com. DNSKEY)
example.com. DS
RRSIG(example.com. DS)
com. DNSKEY
RRSIG(com. DNSKEY)
com. DS
RRSIG(com. DS)
. DNSKEY
RRSIG(. DNSKEY)

ca.example.net. TLSA
RRSIG(ca.example.net. TLSA)
example.net. DNSKEY
RRSIG(example.net. DNSKEY)
example.net. DS
RRSIG(example.net. DS)
net. DNSKEY
RRSIG(net. DNSKEY)
net. DS
RRSIG(net. DS)
```

4. Construction of Serialized Authentication Chains

This section describes a possible procedure for the server to use to build the serialized DNSSEC chain.

When the goal is to perform DANE authentication [RFC6698] of the server's X.509 certificate, the DNS record set to be serialized is a TLSA record set corresponding to the server's domain name.

The domain name of the server MUST be that included in the TLS Server Name Indication extension [RFC6066] when present. If the Server Name Indication extension is not present, or if the server does not recognize the provided name and wishes to proceed with the handshake rather than to abort the connection, the server uses the domain name associated with the server IP address to which the connection has been established.

The TLSA record to be queried is constructed by prepending the _port and _transport labels to the domain name as described in [RFC6698], where "port" is the port number associated with the TLS server. The transport is "tcp" for TLS servers, and "udp" for DTLS servers. The port number label is the left-most label, followed by the transport, followed by the base domain name.

The components of the authentication chain are built by starting at the target record set and its corresponding RRSIG. Then traversing the DNS tree upwards towards the trust anchor zone (normally the DNS root), for each zone cut, the DNSKEY and DS RRsets and their signatures are added. If DNS responses messages contain any domain names utilizing name compression [RFC1035], then they must be uncompressed.

In the future, proposed DNS protocol enhancements, such as the EDNS Chain Query extension [CHAINQUERY] may offer easy ways to obtain all of the chain data in one transaction with an upstream DNSSEC aware recursive server.

5. Caching and Regeneration of the Authentication Chain

DNS records have Time To Live (TTL) parameters, and DNSSEC signatures have validity periods (specifically signature expiration times). After the TLS server constructs the serialized authentication chain, it SHOULD cache and reuse it in multiple TLS connection handshakes. However, it MUST refresh and rebuild the chain as TTLs and signature validity periods dictate. A server implementation could carefully track these parameters and requery component records in the chain correspondingly. Alternatively, it could be configured to rebuild the entire chain at some predefined periodic interval that does not

exceed the DNS TTLs or signature validity periods of the component records in the chain.

6. Verification

A TLS client making use of this specification, and which receives a DNSSEC authentication chain extension from a server, SHOULD use this information to perform DANE authentication of the server certificate. In order to do this, it uses the mechanism specified by the DNSSEC protocol [RFC4035]. This mechanism is sometimes implemented in a DNSSEC validation engine or library.

If the authentication chain is correctly verified, the client then performs DANE authentication of the server according to the DANE TLS protocol [RFC6698], and the additional protocol requirements outlined in [RFC7671].

7. Trust Anchor Maintenance

The trust anchor may change periodically, e.g. when the operator of the trust anchor zone performs a DNSSEC key rollover. Managed key rollovers typically use a process that can be tracked by verifiers allowing them to automatically update their trust anchors, as described in [RFC5011]. TLS clients using this specification are also expected to use such a mechanism to keep their trust anchors updated. Some operating systems may have a system-wide service to maintain and keep the root trust anchor up to date. In such cases, the TLS client application could simply reference that as its trust anchor, periodically checking whether it has changed.

8. Mandating use of this extension

A TLS server certificate MAY mandate the use of this extension by means of the X.509 TLS Feature Extension described in [RFC7633]. This X.509 certificate extension, when populated with the dnssec_chain TLS extension identifier, indicates to the client that the server must deliver the authentication chain when asked to do so. (The X.509 TLS Feature Extension is the same mechanism used to deliver other mandatory signals, such as OCSP "must staple" assertions.)

9. Security Considerations

The security considerations of the normatively referenced RFCs (1035, 4034, 4035, 5246, 6066, 6698, 7633, 7671) all pertain to this extension. Since the server is delivering a chain of DNS records and signatures to the client, it MUST rebuild the chain in accordance with TTL and signature expiration of the chain components as

described in Section 5. TLS clients need roughly accurate time in order to properly authenticate these signatures. This could be achieved by running a time synchronization protocol like NTP [RFC5905] or SNTP [RFC5905], which are already widely used today. TLS clients MUST support a mechanism to track and rollover the trust anchor key, or be able to avail themselves of a service that does this, as described in Section 7.

10. IANA Considerations

This extension requires the registration of a new value in the TLS ExtensionsType registry. The value requested from IANA is 53. If the draft is adopted by the WG, the authors expect to make an early allocation request as specified in [RFC7120].

11. Acknowledgments

Many thanks to Adam Langley for laying the groundwork for this extension. The original idea is his but our acknowledgment in no way implies his endorsement. This document also benefited from discussions with and review from the following people: Viktor Dukhovni, Daniel Kahn Gillmor, Jeff Hodges, Allison Mankin, Patrick McManus, Gowri Visweswaran, Duane Wessels, Nico Williams, and Paul Wouters.

12. References

12.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <http://www.rfc-editor.org/info/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <http://www.rfc-editor.org/info/rfc4035>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <http://www.rfc-editor.org/info/rfc6066>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <http://www.rfc-editor.org/info/rfc6698>.

[RFC7633] Hallam-Baker, P., "X.509v3 Transport Layer Security (TLS) Feature Extension", RFC 7633, DOI 10.17487/RFC7633, October 2015, <http://www.rfc-editor.org/info/rfc7633>.

[RFC7671] Dukhovni, V. and W. Hardaker, "The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance", RFC 7671, DOI 10.17487/RFC7671, October 2015, <http://www.rfc-editor.org/info/rfc7671>.

12.2. Informative References

[RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, RFC 5011, DOI 10.17487/RFC5011, September 2007, <http://www.rfc-editor.org/info/rfc5011>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <http://www.rfc-editor.org/info/rfc5905>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <http://www.rfc-editor.org/info/rfc7120>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <http://www.rfc-editor.org/info/rfc7435>.

[RFC7672] Dukhovni, V. and W. Hardaker, "SMTP Security via Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS)", RFC 7672, DOI 10.17487/RFC7672, October 2015, <http://www.rfc-editor.org/info/rfc7672>.

[AGL] Langley, A., "Serializing DNS Records with DNSSEC Authentication", July 2011, <https://tools.ietf.org/id/draft-agl-dane-serializechain-01.txt>.

[CHAINQUERY]
Wouters, P., "Chain Query Requests in DNS", <https://tools.ietf.org/html/draft-ietf-dnsop-edns-chain-query>.

Appendix A. Pseudocode example

[code goes here]

Appendix B. Test vector

[data go here]

Authors' Addresses

Melinda Shore
No Mountain Software

EMail: melinda.shore@nomountain.net

Richard Barnes
Mozilla

EMail: rlb@ipv.sx

Shumon Huque
Verisign Labs

EMail: shuque@verisign.com

Willem Toorop
NLNet Labs

EMail: willem@nlnetlabs.nl