

TLS
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

M. Shore
No Mountain Software
R. Barnes
Mozilla
S. Huque
Verisign Labs
W. Toorop
NLNet Labs
July 6, 2015

A DANE Record and DNSSEC Authentication Chain Extension for TLS
draft-shore-tls-dnssec-chain-extension-01

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.

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 January 7, 2016.

Copyright Notice

Copyright (c) 2015 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 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.

Table of Contents

1. Requirements Notation	2
2. Introduction	2
3. DNSSEC Authentication Chain Extension	3
3.1. Protocol	3
3.2. DNSSEC Authentication Chain Data	4
4. Construction of Serialized Authentication Chains	5
5. Caching and Regeneration of the Authentication Chain	6
6. Verification	6
7. Trust Anchor Maintenance	7
8. Security Considerations	7
9. IANA Considerations	7
10. Acknowledgments	7
11. Test Vectors	8
12. References	8
12.1. Normative References	8
12.2. Informative References	8
Appendix A. Pseudocode example	10
Appendix B. Test vector	10
Authors' Addresses	10

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 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 records such as TLSA, SMIMEA, etc.

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, and 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 [DANESMTP] should not use this mechanism.

The extension described here allows a TLS client to request in the client hello message that the DNS validation 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 validation 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 [AGL] and it incorporates his ideas and some of his text. It modifies his approach by using DNS wire formats and assumes that in implementation, the serialized DNSSEC object will be prepared by a DNS-specific module and the validation actions on serialized DNSSEC will also be carried out by a DNS-specific module. An appendix (empty in the 00 version) provides a Python code example of interfacing with a DNS-specific module.

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.

[Placeholder: an upcoming revision of this specification will support the ability for the client to include a set of unexpired cached records it possesses, and correspondingly allow the server to return an authentication chain with those records omitted.]

Servers receiving a "dnssec_chain" extension in the client hello 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 represents a sequence of DNS resource record sets, which provide a chain from the DANE record being provided to a trust anchor chosen by the server. The "extension_data" field MUST contain a DNSSEC Authentication Chain encoded in the following form:

```
struct {  
    opaque rrset<0..2^16-1>;  
    opaque rrsig<0..2^16-1>;  
} RRset  
  
RRset AuthenticationChain<0..2^16-1>;
```

Each RRset in the authentication chain encodes an RRset along with a signature on that RRset. The "rrsig" field contains the RDATA for the RRSIG record, defined in [Section 3.1 of RFC 4034 \[RFC4034\]](#). The "rrset" field contains the covered resource records, in the format defined in [Section 3.1.8.1 of RFC 4034 \[RFC4034\]](#):

```
signature = sign(RRSIG_RDATA | RR(1) | RR(2)... )  
  
RR(i) = owner | type | class | TTL | RDATA length | RDATA
```

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

For a DNSKEY RRset, one of the covered DNSKEY RRs MUST be 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 RRset in the authentication chain, representing the trust anchor, SHOULD be omitted. In this case, the client MUST verify that the key tag and owner name in the final RRSIG record correspond to a trust anchor.

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 their corresponding RRSIG signatures):

```
_443._tcp.www.example.com. TLSA
example.com. DNSKEY
example.com. DS
com. DNSKEY
com. DS
. DNSKEY
```

[Some names involving CNAME and DNAMEs may involve multiple branches of the DNS tree. The authors are contemplating enhancements to the AuthenticationChain structure to accommodate these for a future revision of the draft.]

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 aborting 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 and its corresponding RRSIG. Then traversing the DNS tree upwards towards the trust anchor zone (normally the DNS root), for each zone cut, the DS and DNSKEY RRsets and their signatures are added.

In order to meet these formatting requirements, the server must perform some preprocessing on the resource records it receives. It must first compute the uncompressed representation of the RRs, removing DNS name compression [[RFC1035](#)], if present. It then extracts the relevant fields from the resource records and assembles them into an RRset.

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 can cache and reuse it in multiple TLS connection handshakes. However, it should keep track of the TTLs and signature validity periods, and requery the records and rebuild the authentication chain as needed. A server implementation could carefully track these parameters and requery the chain correspondingly. Alternatively, it could be configured to rebuild the 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 [[DANEOPS](#)].

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 up-to-date the root trust anchor. It may be possible for the TLS client application to simply reference that as its trust anchor, periodically checking whether it has changed.

8. Security Considerations

The security considerations of the normatively referenced RFCs (1035, 4034, 4035, 5246, 6066, 6698) all pertain to this extension. Since the server is delivering a chain of DNS records and signatures to the client, it must take care to 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 [RFC4330], which are already widely used today. TLS clients must support a mechanism to track and rollover the trust anchor key as described in Section 7.

9. 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].

10. 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: Allison Mankin, Duane Wessels, Jeff Hodges, Patrick McManus, and Gowri Visweswaran. We are particularly grateful to Viktor Dukhovni for his detailed review.

11. Test Vectors

[TO BE ADDED LATER. THE ORIGINAL CONTENT WAS OBSOLETE.]

12. References

12.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, [RFC 1035](#), November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [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.
- [RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", [RFC 6066](#), January 2011.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", [RFC 6698](#), August 2012.

12.2. Informative References

- [RFC4330] Mills, D., "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", [RFC 4330](#), January 2006.
- [RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, [RFC 5011](#), September 2007.
- [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", [RFC 5905](#), June 2010.
- [RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", [BCP 100](#), [RFC 7120](#), January 2014.

- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", [RFC 7435](#), December 2014.
- [AGL] Langley, A., "Serializing DNS Records with DNSSEC Authentication", <<https://tools.ietf.org/id/draft-agl-dane-serializechain-01.txt>>.
- [DANESMTP] Dukhovni, V. and W. Hardaker, "SMTP Security via opportunistic DANE TLS", <<https://tools.ietf.org/html/draft-ietf-dane-smtp-with-dane-19>>.
- [DANEOPS] Dukhovni, V., "Updates to and Operational Guidance for the DANE Protocol", <<https://tools.ietf.org/html/draft-ietf-dane-ops>>.

[Appendix A.](#) Pseudocode example

[code goes here]

[Appendix B.](#) Test vector

[data go here]

Authors' Addresses

Melinda Shore
No Mountain Software

E-Mail: melinda.shore@nomountain.net

Richard Barnes
Mozilla

E-Mail: rlb@ipv.sx

Shumon Huque
Verisign Labs

E-Mail: shuque@verisign.com

Willem Toorop
NLNet Labs

E-Mail: willem@nlnetlabs.nl