dprive
Internet-Draft
Intended status: Standards Track
Expires: 20 February 2022

B. Schwartz
Google LLC

19 August 2021

Authenticated delegation information using DS records
draft-schwartz-ds-glue-02

Abstract

This draft describes a mechanism for conveying arbitrary authenticated DNS data from a parent nameserver to a recursive resolver as part of a delegation response.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the mailing list (ds@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ds/.

Source for this draft and an issue tracker can be found at https://github.com/bemasc/ds-glue.

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 https://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 20 February 2022.

Table of Contents

1. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Background

The DPRIVE working group has been pursuing designs for authenticated encryption of recursive-to-authoritative communication. Recursive resolvers could enable authenticated encryption most easily and efficiently if they received authenticated information about the target nameserver's configuration during the in-bailiwick delegation that precedes the direct connection. However, there are several obstacles to this.

2.1. Obstacle 1: Authentication

Glue records in DNS referral responses are unauthenticated. Parents do not generally provide RRSIGs for these records in their responses, and resolvers do not expect such signatures to be present. An in-path attacker can modify or remove records in the delegation response without detection.

If the parent zone also implements authenticated encryption, this provides sufficient protection for the glue records, but many important parent zones seem unlikely to implement authenticated encryption in the near future.

2.2. Obstacle 2: Flexibility

Existing nameserver deployments assume that the delegation response includes only a fixed set of existing RR types (NS, A, AAAA, DS, RRSIG, etc.). These systems are slow to upgrade, and the working group would like to be able to begin deploying authenticated encryption without first requiring a significant change in these parents.

3. Proposal

This draft proposes a way to convey a glue RRSet inside a DS record, enabling authenticated delivery of arbitrary RR types as part of the delegation response.

There are three main records or RRSets involved in this process:

* A Source RRSet to be conveyed, which may be of any RR type and anywhere below the zone cut.

* A Virtual DNSKEY Record encapsulating the Source RRSet.

* The DSGLUE Record, a DS record derived from the Virtual DNSKEY Record and published in the parent.

3.1. Encoding

To encode a Source RRSet, a zone operator first transforms it into a Virtual DNSKEY Record as follows:

* Owner Name = The Owner Name of the Source RRSet relative to the child zone apex.

* Flags = 0x0001, i.e. only SEP (bit 15) is set.

* Protocol = 3

* Algorithm = DS Glue (see IANA registration in Section 7)

* Public Key = The following fields, concatenated

  - The RR type (uint16)

  - The RRSet TTL (uint32)

  - For each Source Record in canonical order ([RFC4034], Section 6.3),

    o A length prefix (uint16)

    o The canonicalized RDATA ([RFC4034], Section 6.2).

For example, this Source RRSet:

```
$ORIGIN example.com.
@ 3600 IN NS ns1
       IN NS ns2
       IN NS NS.OTHER.EXAMPLE.
```

would be represented as the following Virtual DNSKEY Record:

```
; Public Key =
; \000\002 ; Type = NS
; \000\000\014\016 ; TTL=3600
; \000\018 \002ns\005other\007example\000 ; Len=18, ns.other.example.
; \000\017 \003ns1\007example\003com\000 ; Len=17, ns1.example.com.
; \000\017 \003ns2\007example\003com\000 ; Len=17, ns2.example.com.

. 300 IN DNSKEY 1 3 $DSGLUE_NUM ( AAIAAA4QABICbnMFb3RoZXIHZXhhbXBsZ
    QAAEQNuczEHZXhhbXBsZQNjb20AABEDbnMyB2V4YW1wbGUDY29tAA== )
```

Note that:

* The NS Source Records are "real" records that appear in authoritative Answers and/or delegation glue, but the DNSKEY record is a "virtual record" because it does not appear in any zone or response (in this form).

* The Virtual DNSKEY Record's owner name is "." because the Source RRSet appears at the zone apex.

* The NS RDATA has been reordered and converted to lowercase as specified by the canonicalization algorithm.

Having constructed a Virtual DNSKEY Record, the DSGLUE Record is constructed as usual, but always using the VERBATIM digest type [I-D.draft-vandijk-dnsop-ds-digest-verbatim]. Thus, the DSGLUE Record's wire format RDATA forms the following concatenation:

```
Key Tag | Algorithm = DSGLUE | Digest Type = VERBATIM | Digest = (
  DNSKEY owner name = name prefix | DNSKEY RDATA = (
    Flags = 1 | Protocol = 3 | Algorithm = DSGLUE | Public Key = (
      RR Type | TTL | Len(1) | RDATA(1) | Len(2) | RDATA(2) | ...
    )
  )
)
```

The DSGLUE record is a real DS record that appears in the usual DS RRSet, whose owner name is the child apex.

QUESTION: Should we skip the virtual DNSKEY record, and construct the fake DS directly? This would save 4-6 bytes per RRSet, but would lose the ability to reuse DNSKEY->DS construction codepaths (unchanged except for a new digest type).

3.2. Interpretation

Upon receiving a delegation response, resolvers implementing this specification SHALL compute the Adjusted Delegation Response as follows:

1. Copy the delegation response.

2. Reverse the encoding process of any DSGLUE records to reconstruct the source RRSets.

3. Add each of these reconstructed RRSets to the Adjusted Delegation Response, replacing any RRSet with the same owner name and type.

Note that a Source RRSet MAY be empty, indicating that there are no records of the corresponding type at this name. After reconstructing an empty Source RRSet, recipients MUST remove any matching RRSets from the Adjusted Delegation Response and any glue cache, and MAY cache the negative result for the indicated TTL.

Resolution then proceeds as usual, using the Adjusted Delegation Response. When processing the DS RRSet, the recipient will verify the DS RRSIGs as usual, and abort the resolution as Bogus if DNSSEC validation fails.

Resolvers that do not implement this specification will ignore the DSGLUE records due to the unrecognized algorithm. Thus, these records are safe to use for both signed and unsigned child zones.

Source Records reconstructed from DSGLUE SHOULD be processed exactly like ordinary unauthenticated glue records. For example, they MAY be cached for use in future delegations but MUST NOT be returned in any responses (c.f. Section 5.4.1 of [RFC2181]).

3.3. Allowed RR types

DSGLUE records are capable of containing any record type. However, the meaning of certain record types (e.g. NSEC) is not yet clear in the DSGLUE context. To avoid ambiguity, child zones MUST only publish DSGLUE records containing RR types that have been registered for use with DSGLUE (Section 7), and recipients MUST ignore DSGLUE records indicating unexpected record types.

Recipients implementing this specification MUST accept the NS, A, and AAAA RR types in DSGLUE. Support for the other allowed RR types is OPTIONAL.

Recipients MUST ignore any unauthenticated TLSA records.

4. Examples

For these examples, the macro "$DSGLUE(prefix, RR type, TTL, [RDATAs])" constructs a DSGLUE DS record as described in Section 3.1.

4.1. Out-of-bailiwick referral

An out-of-bailiwick referral contains only NS records, e.g.

```
$ORIGIN com.
example 3600 IN NS ns1.example.net.
             IN NS ns2.example.net.
```

These Source Records would be encoded in DSGLUE as:

```
$ORIGIN com.
example 3600 IN DS $DSGLUE(., NS, 3600,
    [ns1.example.net., ns2.example.net.])
```

4.2. In-bailiwick referral

An in-bailiwick referral contains NS records and at least one kind of address record.

```
$ORIGIN com.
example    3600 IN NS    ns1.example
                IN NS    ns2.example
ns1.example 600 IN A     192.0.2.1
                IN AAAA  2001:db8::1
ns2.example 600 IN A     192.0.2.2
                IN AAAA  2001:db8::2
```

These records would be encoded in DSGLUE as:

```
$ORIGIN com.
example 600 IN DS $DSGLUE(., NS, 3600, [ns1.example.com.,
                                        ns2.example.com.])
            IN DS $DSGLUE(ns1., A, 600, [192.0.2.1])
            IN DS $DSGLUE(ns1., AAAA, 600, [2001:db8::1])
            IN DS $DSGLUE(ns2., A, 600, [192.0.2.1])
            IN DS $DSGLUE(ns2., AAAA, 600, [2001:db8::2])
```

4.3. In-bailiwick referral without IPv4

Consider a delegation to a nameserver that is only reachable with IPv6:

```
$ORIGIN com.
example    3600 IN NS    ns1.example
ns1.example 600 IN AAAA  2001:db8::1
```

A zone in this configuration can optionally use an empty DSGLUE record to indicate that there is no IPv4 address:

```
$ORIGIN com.
example 600 IN DS $DSGLUE(., NS, 3600, [ns1.example.com.])
            IN DS $DSGLUE(ns1., AAAA, 600, [2001:db8::1])
            IN DS $DSGLUE(ns1., A, 7200, [])
```

This arrangement prevents an adversary from inserting forged A records for ns1.example.com into the delegation response.

Note that this negative answer is treated as glue that only applies during delegation, so A records for ns1.example.com can still be resolved if they exist.

4.4. Delegation with authenticated encryption

Assuming a SVCB-based signaling mechanism similar to [I-D.draft-schwartz-svcb-dns], an in-bailiwick referral with support for authenticated encryption is indicated as follows:

```
$ORIGIN com.
example 600 IN DS $DSGLUE(., NS, 3600, [ns1.example.com.])
            IN DS $DSGLUE(ns1., A, 600, [192.0.2.1])
            IN DS $DSGLUE(ns1., AAAA, 600, [2001:db8::1])
            IN DS $DSGLUE(_dns.ns1., SVCB, 3600,
                          [1 ns1.example.com. alpn=dot])
```

4.4.1. Disabling DANE

Resolvers check whether a nameserver supports DANE by resolving a TLSA record during the delegation process (Section 6.4). However, this adds unnecessary latency to the delegation if the nameserver does not implement DANE. As an optimization, such nameservers can add an empty DSGLUE RRSet to indicate that there is no such TLSA record, e.g.:

```
IN DS $DSGLUE(_853._tcp.ns1., TLSA, 7200, [])
```

5. Security Considerations

Resolvers that process DSGLUE MUST perform DNSSEC validation.

Source Records published as DSGLUE have owner names within the child zone, but are signed only by the parent. This makes them fully authenticated, but provides different cryptographic guarantees than a direct signature by the child. For example, these records might not appear in any key use logs maintained by the child.

6. Operational Considerations

6.1. Compatibility with existing resolvers

Resolver support for DSGLUE is OPTIONAL, so child zones MUST continue to place ordinary NS, A, and AAAA records in the parent zone as needed for non-DSGLUE resolution.

6.2. Publishing DSGLUE records

In order for the child to publish DSGLUE records, the parent must allow the child to publish arbitrary DS records or have specific support for this specification.

If the parent supports CDS [RFC8078], child zones MAY use CDS to push DSGLUE records into the parent. Note that CDNSKEY records cannot be used, because (1) the child cannot publish CDNSKEY records with the required owner name and (2) the child cannot guarantee that the parent will use the VERBATIM digest to produce the DS record.

Child zones SHOULD publish all Source Records as ordinary records of the specified type at the indicated owner name, in order to enable revalidation [I-D.draft-ietf-dnsop-ns-revalidation] and simplify debugging.

6.3. Referral response size

When records are present in both ordinary glue and DSGLUE, the response size is approximately doubled. This could cause performance issues due to response truncation when the initial query is over UDP.

6.4. PKI and DANE for Authenticated Encryption

TODO: Move some of this text into a different draft.

Nameservers supporting authenticated encryption MAY indicate any DANE mode, or none at all.

As an optimization, nameservers using DANE MAY place a TLSA record in the DSGLUE to avoid the latency of a TLSA lookup during delegation. However, child zones should be aware that this adds complexity and delay to the process of TLSA key rotation.

QUESTION: Should we recommend for or against including nonempty TLSA in DSGLUE? If CDS-like update mechanisms work well, and ADoT-DANE is widely deployed, this could warrant a positive recommendation. Conversely, if rotation is error-prone, and ADoT-DANE is rare, a negative recommendation might be better.

Nameservers that support PKI-based authentication but not DANE SHOULD deny the TLSA RRSet in the DSGLUE, as shown in Section 4.4.1, to avoid an unnecessary delay.

Resolvers that support authenticated encryption MAY implement support for PKI-based authentication, DANE, or both. PKI-only resolvers MUST nonetheless resolve TLSA records, and MUST NOT require authentication

if the DANE mode is DANE-TA(2) or DANE-EE(3) [RFC7671]. DANE-only resolvers MUST NOT require authentication if the TLSA record does not exist.

7. IANA Considerations

IANA is requested to add a new entry to the DNS Security Algorithm Numbers registry:

Number	Description	Mnemonic	Zone Signing	Trans. Sec.	Reference
$DSGLUE_NUM	Authenticated Glue	DSGLUE	N	?	(This document)

Table 1

IANA is requested to open a new registry named "Authenticated Glue Allowed Record Types", with a policy of "Standards Action" and the following fields:

* Record Type: The name of a registered DNS record type

* Interpretation Reference: The standards document defining how to interpret this RR type in the Authenticated Glue context.

The initial contents are as follows:

Record Type	Interpretation Reference
NS	(This document)
A	(This document)
AAAA	(This document)
SVCB	(This document)
TLSA	(This document)

Table 2

8. References

8.1. Normative References

[I-D.draft-vandijk-dnsop-ds-digest-verbatim]
Dijk, P. V., "The VERBATIM Digest Algorithm for DS records", Work in Progress, Internet-Draft, draft-vandijk-dnsop-ds-digest-verbatim-01, 10 August 2021, <https://datatracker.ietf.org/doc/html/draft-vandijk-dnsop-ds-digest-verbatim-01>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/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, <https://www.rfc-editor.org/rfc/rfc4034>.

[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, <https://www.rfc-editor.org/rfc/rfc7671>.

[RFC8078] Gudmundsson, O. and P. Wouters, "Managing DS Records from the Parent via CDS/CDNSKEY", RFC 8078, DOI 10.17487/RFC8078, March 2017, <https://www.rfc-editor.org/rfc/rfc8078>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

8.2. Informative References

[I-D.draft-ietf-dnsop-ns-revalidation]
Huque, S., Vixie, P., and R. Dolmans, "Delegation Revalidation by DNS Resolvers", Work in Progress, Internet-Draft, draft-ietf-dnsop-ns-revalidation-01, 12 July 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-ns-revalidation-01>.

[I-D.draft-schwartz-svcb-dns]
Schwartz, B., "Service Binding Mapping for DNS Servers", Work in Progress, Internet-Draft, draft-schwartz-svcb-dns-04, 26 July 2021, <https://datatracker.ietf.org/doc/html/draft-schwartz-svcb-dns-04>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/rfc/rfc2181>.

Acknowledgments

Thanks to Paul Hoffman, Ilari Liusvaara, Puneet Sood, and Alexandar Mayrhofer for detailed comments.

Author's Address

Benjamin Schwartz
Google LLC

Email: bemasc@google.com