

Independent Submission
Internet-Draft
Intended status: Best Current Practice
Expires: December 10, 2020

R. Arends
E. Lewis
ICANN
June 08, 2020

Top-level Domains for Private Internets
draft-arends-private-use-tld-02

Abstract

There are no defined private-use namespaces in the Domain Name System (DNS). For a domain name to be considered private-use, it needs to be future-proof in that its top-level domain will never be delegated from the root zone. The lack of a private-use namespace has led to locally configured namespaces with a top-level domain that is not future proof.

The DNS needs an equivalent of the facilities provided by BCP 5 (RFC 1918) for private internets, i.e. a range of short, semantic-free top-level domains that can be used in private internets without the risk of being globally delegated from the root zone.

The ISO 3166 standard is used for the definition of eligible designations for country-code top-level Domains. This standard is maintained by the ISO 3166 Maintenance Agency. The ISO 3166 standard includes a set of user-assigned code elements that can be used by those who need to add further names to their local applications.

Because of the rules set out by ISO in their standard, it is extremely unlikely that these user-assigned code elements would ever conflict with delegations in the root zone under current practices. This document explicitly reserves these code elements to be safely used as top-level domains for private DNS resolution.

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 December 10, 2020.

Copyright Notice

Copyright (c) 2020 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. Introduction	2
2. The ISO 3166-1 alpha-2 and Two-Letter Top-Level Domains . . .	4
3. ISO 3166-1 alpha-2 User-assigned Code Elements	4
4. Examples of Current Uses of the User-assigned Code Elements.	5
5. Private-use top-level Domains	8
6. IANA Considerations	9
7. Security Considerations	9
8. Acknowledgements	9
9. Informative References	9
Authors' Addresses	12

1. Introduction

In private networks where a hostname has no utility in the global namespace, it is convenient to have a private-use namespace. Such deployments could theoretically use sub-domains of a domain registered for the specific hosting entity, though not all such configurations have such a domain available. When the hostname is solely used in a private network, it is not necessary that it resolves globally.

Another situation is where applications use identifiers that are similar in appearance to domain names, and may be interpreted by software as domain names, but are not intended to use the global DNS resolution service. Using a private-use namespace helps guard against conflicts with the global DNS resolution system.

Note that a private-use namespace is not a subset of a registered special use namespace [IANA-Special]. There is no facility to register a specific label using the process defined in [RFC6761]. The process in RFC 6761 requires that a label has some kind of special handling in order to be considered special. A private-use namespace can be considered special on a policy level, but not on a technical or protocol level.

Many protocols outside the DNS have a defined set of elements for private use, or an identifier that indicates private use, such as "X-headers" MIME types [RFC2045], addresses for private internets [BCP-5], the "x-" sub-tag in private-use language tags [BCP-47], private-use Autonomous System Numbers [BCP-6], and private-use DNS RRTypes and RCODEs [BCP-42].

There is currently no such facility for the DNS namespace. A user is required to resort to registering a globally unique domain where a locally unique domain would suffice, or may configure a domain name that is not currently delegated from the root zone. Additionally, there are plenty of examples of device vendors that ship networking devices with a default setting for DHCP [RFC2131] option 15 (domain name) [RFC2132], containing a top-level domain that is believed to not be delegated in the root zone.

In practice, the lack of a private-use namespace facility has led to the deployment of arbitrary, unregistered, semantically meaningful top-level domains, such as ".home", ".dhcp", ".lan", ".localdomain", ".internal", ".dlink", ".ip" and ".corp" [ITHI]. These examples of locally configured strings are derived from traffic to the ICANN Managed Root Servers [IMRS] and are part of the most popular observed query names [BCP-219].

While these commonly chosen strings currently do not exist in the root zone, there is no guarantee that these strings will not be delegated in the root zone in the future. Therefore, there is no guarantee that the local use of these strings (or other strings that might be chosen for private use) will be stable, safe, and secure. Similarly, there is no guarantee that any of these strings will be deemed special-use via an application according to [RFC6761].

There are many uses for private-use names. It is not feasible to assign a semantically meaningful, relatively short top-level label to each individual private-use of a namespace in multiple languages. Similar to "X-headers" MIME types, and analogous in concept to address allocation for private internets, this document defines a range of abstract, two-letter labels that are aligned with the user-assigned two-letter code elements in the ISO 3166-1 alpha-2 [ISO3166-1] standard.

2. The ISO 3166-1 alpha-2 and Two-Letter Top-Level Domains

IANA's practice of governing the delegation of ASCII two-letter domain names in the DNS [STD13] root zone is to align it with assignment of two-letter (known as "alpha-2") code elements in the ISO 3166-1 standard [ISO3166-1]. The ISO 3166-1 standard contains many categories of code elements, with the "officially assigned" and some "exceptionally reserved" code elements being used in the DNS to represent entities as country-code top-level domains (ccTLDs) [RFC1591]. The interrelationship is documented in "ICANN and the ISO, A Common Interest in ISO Standard 3166" [ICANNISO].

In addition to the assigned, available, and reserved code elements, there are code elements designated as "user-assigned". The intent of user-assigned code elements is to provide the user with a code element when no other code element satisfies the intended use.

3. ISO 3166-1 alpha-2 User-assigned Code Elements

The ISO 3166-1 standard states in section 5.2:

"In addition, exactly 42 alpha-2 code elements are not used in the ISO 3166, AA, QM to QZ, XA to XZ, ZZ."

And explains in clause 8.1 "Special Provisions":

"Users sometimes need to extend or alter the use of country-code elements for special purposes. The following provisions give guidance for meeting such needs within the framework of this part of ISO 3166. "

And finally, clause 8.1.3 "User assigned code element":

"If users need code elements to represent country names not included in this part of ISO 3166, the series of letters AA, QM to QZ, XA to XZ, and ZZ, and the series AAA to AAZ, QMA to QZZ, XAA to XZZ, and ZZA to ZZZ respectively and the series of numbers 900 to 999 are available. NOTE Users are advised that the above series of codes are not universals, those code elements are not compatible between different entities."

As shown above, the ISO 3166-1 user-assigned alpha-2 code elements are defined to be AA, QM to QZ, XA to XZ, and ZZ. The ranges ("to") are alphabetic and contain only characters in the US-ASCII definition [STD80].

It is unlikely that the user-assigned range will change.

4. Examples of Current Uses of the User-assigned Code Elements.

Using code elements to represent an entity other than a country name may appear to deviate from the intended use of the ISO 3166-1 standard. However, many organizations, including the IETF and the ISO, have used the user-assigned range to represent entities other than country names. The following list is not exhaustive but illustrates the wide variety of current uses of codes within the ISO 3166-1 user-assigned alpha-2 range.

- o The International Standard Recording Code (ISRC) [ISO3901] uses code element "ZZ" from the User-assigned range for direct registrants independent of any country.
- o The ISO Currency Codes standard [ISO4217] uses code elements "XA" to "XZ" from the user-assigned range for transactions and precious metals.
- o International Securities Identification Numbers [ISO6166] uses the following code elements from the user-assigned range:

QS: internally used by Euroclear France

QT: internally used in Switzerland

QW: internally used in WM Datenservice Germany for historical data

XA: CUSIP Global Services substitute agencies

XB: NSD Russia substitute agencies

XC: WM Datenservice Germany substitute agencies

XD: SIX Telekurs substitute agencies

XF: internally assigned, non-unique numbers

XS: Euroclear and Clearstream international securities

- o The International Civil Aviation Organization [ICAO] Machine Readable Travel Documents standard uses code element "ZZ" from the user-assigned range for UN travel documents.

- o The World Intellectual Property Organization [WIPO] Standard 3 uses the following code elements from the user-assigned range:

QZ: Community Plant Variety Office (European Union) (CPVO).

XN: Nordic Patent Institute (NPI).

XU: International Union for the Protection of New Varieties of Plants (UPOV).

XV: Visegrad Patent Institute (VPI)

XX: recommended to refer to unknown states, other entities or organizations.

- o The United Nations Code for Trade and Transport Locations [UNLOCODE] uses the code element "XZ" from the user-assigned range for international waters in accordance with ISO 3166-1 clause 8.1.3:

"3.2.5 In cases where no ISO 3166 country-code element is available, e.g. installations in international waters or international cooperation zones, the code element "XZ", available for user assignment in accordance with clause 8.1.3 of ISO 3166-1/1997, will be used."

- o The World Bank Country API [WORLDBANK] uses the following code elements from the User-assigned range:

XC: Euro area

XD: High income

XE: Heavily indebted poor countries (HIPC)

XF: International Bank for Reconstruction and Development

XH: Blend

XI: International Development Association

XJ: Latin America and Caribbean (excluding high income)

XL: Least developed countries: UN classification

XM: Low income

XN: Lower middle income

XO: Low & middle income

XP: Middle income

XQ: Middle East & North Africa (excluding high income)

XT: Upper middle income

XU: North America

XX: Not classified

XY: Not Classified

- o The Interpol Implementation data format for the interchange of fingerprint, facial & scar-mark-tattoo information [INTERPOL] uses code element "ZZ" from the user-assigned range as follows: Destination Agency Identifier "ZZ/ALL" is reserved for transactions which shall be distributed by INTERPOL AFIS to all INTERPOL member states."
- o The Certificate Authority Browser Forum Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates [CABForum] states that if a country is not represented by an official ISO 3166-1 alpha-2 country-code, the CA may specify the user-assigned code element "XX" to indicate that an official code element has not been assigned.
- o The UNICODE Common Locale Data Repository (CLDR) [UNICODE] version 37 uses the following code elements from the user-assigned range:

QO: Outlying Oceania; countries in Oceania that do not have a subcontinent.

XA: Pseudo-Accents; special code indicating derived testing locale with English + added accents and lengthened.

XB: Pseudo-Bidi; special code indicating derived testing locale with forced RTL English.

ZZ: Unknown Region; used in APIs or as a replacement for invalid code.

- o The IETF Best Current Practice 47 [BCP-47] contains a section and examples dedicated to private-use subtags, using code elements from the user-assigned range:

"For example, the region subtags 'AA', 'ZZ', and those in the ranges 'QM'-'QZ' and 'XA'-'XZ' (derived from the ISO 3166-1 alpha-2 private use codes) can be used to form a language tag. A tag such as "zh-Hans-XQ" conveys a great deal of public, interchangeable information about the language material"

- o The IETF Proposed Standard "Internationalized Domain Names for Applications" [RFC5890] uses the XN-- prefix. The method that was used to decide on the prefix was explained in an email from the IANA to the IETF IDN Working Group list [XNIDN]:

"The following steps will be used to select the two-character code:

The code will be selected from among a subset of the entries on the ISO 3166-1, clause 8.1.3 User-assigned alpha-2 code elements: AA, QM to QZ, XA to XZ, and ZZ. The selection is limited to these codes because of the following:

The use of ISO 3166-1 User-assigned elements removes the possibility that the code will duplicate a present or future ccTLD code."

5. Private-use top-level Domains

The user-assigned classification of these code elements in the ISO 3166-1 alpha-2 standard allows for the assumption that these code elements will not risk delegation as country-code top-level Domains through future assignments to represent a country or territory. To quote [XNIDN]:

"The use of ISO 3166-1 User-assigned elements removes the possibility that the code will duplicate a present or future ccTLD code."

Using these code elements as top-level domains for the purpose of private-use TLDs is in line with the intended use of these code elements and follows the many examples of other standards and protocols. Furthermore, they are short and free of any semantic meaning.

This document does not recommend any specific ISO 3166-1 alpha-2 user-assigned code as a private use, but instead proposes that any of them can be used by a network or application for private use. That is, there is no attempt to choose just one of the ISO 3166-1 Alpha-2 user-assigned codes for use as private-use TLDs, just as other organizations use multiple user-assigned codes for many internal purposes.

Note that there may be software that treats labels beginning with XN differently due to the use of the XN-- prefix in internationalized domain names [RFC5890].

6. IANA Considerations

This document makes the observation that the policy of assigning ccTLD labels is to align with the ISO-3166-1 alpha-2 standard [RFC1591], which includes user-assigned code elements that will never be assigned to a territory [ISO3166-1]. This is then consistent with existing policies that those user-assigned codes will never be delegated from the DNS root zone and, for that reason, will never give rise to collisions with any future new TLD.

7. Security Considerations

Use of private-use identifiers of any sort almost always results in unexpected collisions in practice. This has repeatedly been shown for private-use addresses, private-use identifiers (such as "x-headers") and private-use names in the DNS. These unexpected collisions can easily have security ramifications that are well beyond what the user understands.

8. Acknowledgements

This document is based on an earlier draft by Ed Lewis. David Conrad, Paul Hoffman, Sion Lloyd, Alain Durand, Jaap Akkerhuis, Kal Feher, Andrew Sullivan, Joe Abley, Petr Spacek, Patrick Mevzek and Kim Davies have played a role.

9. Informative References

- [BCP-219] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [BCP-42] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", BCP 42, RFC 6895, DOI 10.17487/RFC6895, April 2013, <<https://www.rfc-editor.org/info/rfc6895>>.
- [BCP-47] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <<https://www.rfc-editor.org/info/rfc5646>>.
- [BCP-5] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <<https://www.rfc-editor.org/info/rfc1918>>.
- [BCP-6] Mitchell, J., "Autonomous System (AS) Reservation for Private Use", BCP 6, RFC 6996, DOI 10.17487/RFC6996, July 2013, <<https://www.rfc-editor.org/info/rfc6996>>.

- [CABForum] "CA/Browser Forum Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates, version 1.6.9", March 2020, <<https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.6.9.pdf>>.
- [IANA-Special] "Special-Use Domain Names", n.d., <<https://www.iana.org/assignments/special-use-domain-names/special-use-domain-names.xhtml>>.
- [ICANNISO] "ICANN and the International Organization for Standardization (ISO)", n.d., <<https://www.icann.org/resources/pages/icann-iso-3166-2012-05-09-en>>.
- [ICAO] "International Civil Aviation Organization, Machine Readable Travel Documents, Part 3; Specifications Common to all MRTDs", n.d., <https://www.icao.int/publications/Documents/9303_p3_cons_en.pdf>.
- [IMRS] "ICANN Managed Root Server", n.d., <<https://www.dns.icann.org/imrs/>>.
- [INTERPOL] "Interpol Implementation data format for the interchange of fingerprint, facial & smt information", n.d., <<https://www.interpol.int/en/How-we-work/Forensics/Fingerprints>>.
- [ISO3166-1] "ISO 3166-1:2013 Codes for the representation of names of countries and their subdivisions - Part 1: Country codes", 2013, <<https://www.iso.org/standard/63545.html>>.
- [ISO3901] "Information and documentation -- International Standard Recording Code (ISRC)", n.d., <<https://www.iso.org/standard/64817.html>>.
- [ISO4217] "ISO 4217; Codes for the representation of currencies and funds", n.d., <<https://www.iso.org/iso-4217-currency-codes.html>>.
- [ISO6166] "Securities and related financial instruments -- International securities identification numbering system (ISIN)", n.d., <<https://www.iso.org/standard/44811.html>>.

- [ITHI] "ICANN's Identifier Technology Health Indicator; Queries to frequently leaked strings", n.d.,
<<https://ithi.research.icann.org/graph-m3.html#M332>>.
- [RFC1591] Postel, J., "Domain Name System Structure and Delegation", RFC 1591, DOI 10.17487/RFC1591, March 1994,
<<https://www.rfc-editor.org/info/rfc1591>>.
- [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996,
<<https://www.rfc-editor.org/info/rfc2045>>.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997,
<<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997,
<<https://www.rfc-editor.org/info/rfc2132>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010,
<<https://www.rfc-editor.org/info/rfc5890>>.
- [RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013,
<<https://www.rfc-editor.org/info/rfc6761>>.
- [STD13] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987,
<<https://www.rfc-editor.org/info/rfc1034>>.
- [STD80] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969,
<<https://www.rfc-editor.org/info/rfc20>>.
- [UNICODE] "CLDRv37 - Unicode Common Locale Data Repository version 37", April 2020,
<<http://cldr.unicode.org/index/downloads/cldr-37>>.
- [UNLOCODE] "United Nations Code for Trade and Transport Locations; UN/LOCODE Manual", n.d.,
<https://www.unece.org/fileadmin/DAM/cefact/locode/UNLOCODE_Manual.pdf>.

[WIPO] "World Intellectual Property Organization; Recommended standard on two-letter codes for the representation of states, other entities and intergovernmental organizations.", n.d., <<https://www.wipo.int/export/sites/www/standards/en/pdf/03-03-01.pdf>>.

[WORLDBANK] "Worldbank API V2 Country API", n.d..

[XNIDN] "Results of IANA Selection of IDNA Prefix", February 2003, <<https://psg.com/~randy/lists/iesg/2003/msg01081.html>>.

Authors' Addresses

Roy Arends
ICANN

Email: roy.arends@icann.org

Ed Lewis
ICANN

Email: ed.lewis@icann.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 21, 2020

A. Brotman
Comcast, Inc
S. Farrell
Trinity College Dublin
September 18, 2019

Related Domains By DNS
draft-brotman-rdbd-03

Abstract

This document describes a mechanism by which a DNS domain can publicly document the existence or absence of a relationship with a different domain, called "Related Domains By DNS", or "RDBD."

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 March 21, 2020.

Copyright Notice

Copyright (c) 2019 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 (<https://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. Introduction	2
1.1. Use-Cases	3
1.2. Terminology	3
2. New Resource Record Types	4
2.1. RDBDKEY Resource Record Definition	4
2.2. RDBD Resource Record Definition	5
3. RDBD processing	7
4. Use-cases for Signatures	8
4.1. Many-to-one Use-Case	8
4.2. Extending DNSSEC	8
5. Security Considerations	9
5.1. Efficiency of signatures	9
5.2. DNSSEC	9
5.3. Lookup Loops	9
6. IANA Considerations	10
7. Acknowledgements	10
8. References	10
8.1. Normative References	10
8.2. Informative References	11
Appendix A. Implementation (and Toy Deployment:-) Status	11
Appendix B. Examples	11
Appendix C. Possible dig output...	14
Appendix D. Changes and Open Issues	15
D.1. Changes from -02 to -03	15
D.2. Changes from -01 to -02	16
D.3. Changes from -00 to -01	16
Authors' Addresses	16

1. Introduction

Determining relationships between DNS domains can be one of the more difficult investigations on the Internet. It is typical to see something such as "example.com" and "dept-example.com" and be unsure if there is an actual relationship between those two domains, or if one might be an attacker attempting to impersonate the other. In some cases, anecdotal evidence from the DNS or WHOIS/RDAP may be sufficient. However, service providers of various kinds may err on the side of caution and treat one of the domains as untrustworthy or abusive if it is not clear that the two domains are in fact related. This specification provides a way for one domain to explicitly document, or disavow, relationships with other domains, utilizing DNS records.

It is not a goal of this specification to provide a high-level of assurance as to whether or not two domains are definitely related, nor to provide fine-grained detail about the kinds of relationships

that may exist between domains. However, the mechanism defined here is extensible in a way that should allow use-cases calling for such declarations to be handled later.

1.1. Use-Cases

The use cases for this include:

- o where an organisation has names below different ccTLDs, and would like to allow others to correlate their ownership more easily, consider "example.de" and "example.ie" registered by regional offices of the same company;
- o following an acquisition, a domain holder might want to indicate that example.net is now related to example.com in order to make a later migration easier;
- o when doing Internet surveys, we should be able to provide more accurate results if we have information as to which domains are, or are not, related;
- o a domain holder may wish to declare that no relationship exists with some other domain, for example "good.example" may want to declare that it is not associated with "g00d.example" if the latter is currently being used in some cousin-domain style attack in which case, it is more likely that there can be a larger list of names (compared to the "positive" use-cases) for which there is a desire to disavow a relationship.

[[Discussion of this draft is taking place on the dnsop@ietf.org mailing list. Previously, discussion was on the dbound@ietf.org list. There's a github repo for this draft at <<https://github.com/abrotman/related-domains-by-dns>> - issues and PRs are welcome there.]]

1.2. Terminology

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 [RFC2119].

The following terms are used throughout this document:

- o Relating-domain: this refers to the domain that is declaring a relationship exists. (This was called the "parent/primary" in -00).

- o Related-domain: This refers to the domain that is referenced by the Relating-domain, such as "dept-example.com". (This was called the "secondary" in -00.)

2. New Resource Record Types

We define a resource record type (RDBD) that can declare, or disavow, a relationship. RDBD also includes an optional digital signature mechanism that can somewhat improve the level of assurance with which an RDBD declaration can be handled. This mechanism is partly modelled on how DKIM [RFC6376] handles public keys and signatures - a public key is hosted at the Relating-domain (e.g., "club.example.com"), using an RDBDKEY resource record, and the RDBD record of the Related-domain (e.g., "member.example.com") can contain a signature (verifiable with the "club.example.com" public key) over the text representation ('A-label') of the two names (plus a couple of other inputs).

2.1. RDBDKEY Resource Record Definition

The RDBDKEY record is published at the apex of the Relating-domain zone.

The wire and presentation format of the RDBDKEY resource record is identical to the DNSKEY record. [RFC4034]

[[All going well, at some point we'll be able to say...]] IANA has allocated RR code TBD for the RDBDKEY resource record via Expert Review. [[In the meantime we're experimenting using 0xffa8, which is decimal 65448, from the experimental RR code range, for the RDBDKEY resource record.]]

The RDBDKEY RR uses the same registries as DNSKEY for its fields. (This follows the precedent set for CDNSKEY in [RFC7344].)

No special processing is performed by authoritative servers or by resolvers, when serving or resolving. For all practical purposes, RDBDKEY is a regular RR type.

The flags field of RDBDKEY records MUST be zero. [[Is that correct/ok?]]

There can be multiple occurrences of the RDBDKEY resource record in the same zone.

2.2. RDBD Resource Record Definition

To declare a relationship exists an RDBD resource record is published at the apex of the Related-domain zone.

To disavow a relationship an RDBD resource record is published at the apex of the Relating-domain zone.

[[All going well, at some point we'll be able to say...]] IANA has allocated RR code TBD for the RDBD resource record via Expert Review. [[In the meantime we're experimenting using 0xffa3, which is decimal 65443, from the experimental RR code range, for the RDBD resource record.]]

The RDBD RR is class independent.

The RDBD RR has no special Time to Live (TTL) requirements.

There can be multiple occurrences of the RDBD resource record in the same zone.

RDBD relationships are uni-directional. If bi-directional relationships exist, then both domains can publish RDBD RRs and optionally sign those.

The wire format for an RDBD RDATA consists of a two octet rdbd-tag, a domain name or URL, and the optional signature fields which are: a two-octet key-tag, a one-octet signature algorithm, and the digital signature bits.

```

          1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               |                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
/                               domain name or URL                               /
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   key-tag   |   sig-alg   |                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
/                               signature                               /
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

We define two possible values for the rdbd-tag in this specification, later specifications can define new rdbd-tag values:

- o 0: states that no relationship exists between the domains
- o 1: states that some relationship exists between the domains

The domain name field contains either a single domain name, or an HTTPS URL. In the latter case, successfully de-referencing that URL is expected to result in a JSON object that contains a list of domain names, such as is shown in the figure below.

```
[
  "example.com",
  "example.net",
  "foo.example"
]
```

If an optional signature is included, the sig-alg field MUST contain the signature algorithm used, with the same values used as would be used in an RRSIG. The key-tag MUST match the RDBDKEY RR value for the corresponding public key, and is calculated as defined in [RFC4034] appendix B.

If the optional signature is omitted, then the presentation form of the key-tag, sig-alg and signature fields MAY be omitted. If not omitted then the sig-alg and key-tag fields MUST be zero and the signature field MUST be a an empty string. [[Is that the right way to have optional fields in prsentation syntax for RRs?]]

The input to signing ("to-be-signed" data) is the concatenation of the following linefeed-separated (where linefeed has the value '0x0a') lines:

```
relating=<Relating-domain name>
related=<Related-domain name or URL>
rdbd-tag=<rdbd-tag value>
key-tag=<key-tag>
sig-alg=<sig-alg>
```

The Relating-domain and Related-domain values MUST be the 'A-label' representation of these names. The trailing "." representing the DNS root MUST NOT be included in the to-be-signed data, so a Relating-domain value above might be "example.com" but "example.com." MUST NOT be used as input to signing.

The rdbd-tag and key-tag and sig-alg fields MUST be in decimal with leading zeros omitted.

A linefeed MUST be included after the "sig-alg" value in the last line.

[[Presentation syntax and to-be-signed details are very liable to change.]]

See the examples in the Appendix for further details.

3. RDBD processing

- o If multiple RDBD records exist with conflicting "rdbd-tag" values, those RDBD records SHOULD be ignored.
- o If an RDBD record has an invalid or undocumented "rdbd-tag", that RDBD record SHOULD be ignored.
- o The document being referenced by a URL within an RDBD record MUST be a well-formed JSON [RFC8259] document. If the document does not validate as a JSON document, the contents of the document SHOULD be ignored. There is no defined maximum size for these documents, but a referring site ought be considerate of the retrieving entity's resources.
- o When retrieving the document via HTTPS, the certificate presented MUST properly validate. If the certificate fails to validate, the retrieving entity SHOULD ignore the contents of the file located at that resource.
- o Normal HTTP processing rules apply when de-referencing a URL found in an RDBD record, for example, a site may employ HTTP redirection.
- o Consumers of RDBD RRs MAY support signature verification. They MUST be able to parse/process unsigned or signed RDBD RRs even if they cannot cryptographically verify signatures.
- o Implementations producing RDBD RRs SHOULD support optional signing of those and production of RDBDKEY RRs.
- o Implementations of this specification that support signing or verifying signatures MUST support use of RSA with SHA256 (sig-alg==8) with at least 2048 bit RSA keys. [RFC5702]
- o RSA keys MUST use a 2048 bit or longer modulus.
- o Implementations of this specification that support signing or verifying signatures SHOULD support use of Ed25519 (sig-alg==15). [RFC8080][RFC8032]

- o A validated signature is solely meant to be additional evidence that the relevant domains are related, or that one disavows such a relationship.

4. Use-cases for Signatures

[[The signature mechanism is pretty complex, relative to anything else here, so it might be considered as an at-risk feature.]]

We see two possibly interesting use-cases for the signature mechanism defined here. They are not mutually exclusive.

4.1. Many-to-one Use-Case

If a bi-directional relationship exists between one Relating-domain and many Related-domains and the signature scheme is not used, then making the many required changes to the Relating-domain zone could be onerous. Instead, the signature mechanism allows one to publish a stable value (the RDBDKEY) once in the Relating-domain. Each Related-domain can then also publish a stable value (the RDBD RR with a signature) where the signature provides confirmation that both domains are involved in declarating the relationship.

This scenario also makes sense if the relationship (represented by the rdbd-tag) between the domains is inherently directional, for example, if the relationship between the Related-domains and Relating-domain is akin to a membership relationship.

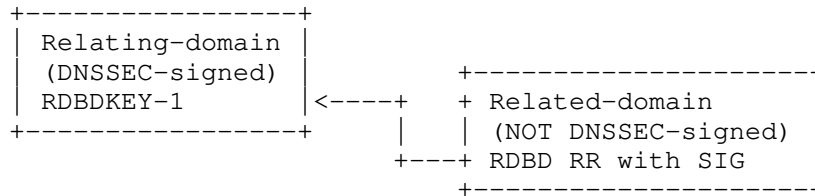
4.2. Extending DNSSEC

If the Relating-domain and Related-domain zones are both DNSSEC-signed, then the signature mechanism defined here adds almost no value and so is unlikely to be worth deploying in that it provides no additional cryptographic security (though the many-to-one advantage could still apply). If neither zone is DNSSEC-signed, then again, there may be little value in deploying RDBD signatures.

The minimal value that remains in either such case, is that if a client has acquired and cached RDBDKEY values in some secure manner, then the RDBD signatures do offer some benefit. However, at this point it seems fairly unlikely that RDBDKEY values will be acquired and cached via some secure out-of-band mechanisms, so we do not expect much deployment of RDBD signatures in either the full-DNSSEC or no-DNSSEC cases.

However, where the Relating-domain's zone is DNSSEC-signed, but the Related-domain's zone is not DNSSEC signed, then the RDBD signatures

do provide value, in essence by extending DNSSEC "sideways" to the Related-domain. The figure below illustrates this situation.



Extending DNSSEC use-case for RDBD signatures

5. Security Considerations

5.1. Efficiency of signatures

The optional signature mechanism defined here offers no protection against an active attack if both the RDBD and RDBDKEY values are accessed via an untrusted path.

5.2. DNSSEC

RDBD does not require DNSSEC. Without DNSSEC it is possible for an attacker to falsify DNS query responses for someone investigating a relationship. Conversely, an attacker could delete the response that would normally demonstrate the relationship, causing the investigating party to believe there is no link between the two domains. An attacker could also replay an old RDBD value that is actually no longer published in the DNS by the Related-domain.

Deploying signed records with DNSSEC should allow for detection of these kinds of attack.

5.3. Lookup Loops

A bad actor could create a loop of relationships, such as a.example->b.example->c.example->a.example or similar. Automated systems SHOULD protect against such loops. For example, only performing a configured number of lookups from the first domain. Publishers of RDBD records SHOULD attempt to keep links direct and so that only the fewest number of lookups are needed, but it is understood this may not always be possible.

6. IANA Considerations

This document introduces two new DNS RR types, RDBD and RDBDKEY. [[Codepoints for those are not yet allocated by IANA, nor have codepoints been requested so far.]]

[[New rdbd-tag value handling will need to be defined if we keep that field. Maybe something like: 0-255: RFC required; 256-1023: reserved; 1024-2047: Private use; 2048-65535: FCFS. It will also likely be useful to define a string representation for each registered rdbd-tag value, e.g. perhaps "UNRELATED" for rdbd-tag value 0, and "RELATED" for rdbd-tag value 1, so that tools displaying RDBD information can be consistent.]]

7. Acknowledgements

Thanks to all who commented on this on the dbound and other lists, in particular to the following who provided comments that caused us to change the draft: Bob Harold, John Levine, Pete Resnick, Andrew Sullivan, Tim Wisinski, Suzanne Woolf, Joe St. Sauver, and Paul Wouters. (We're not implying any of these fine folks actually like this draft btw, but we did change it because of their comments:-) Apologies to anyone we missed, just let us know and we'll add your name here.

8. References

8.1. Normative References

- [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/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, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC5702] Jansen, J., "Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC", RFC 5702, DOI 10.17487/RFC5702, October 2009, <<https://www.rfc-editor.org/info/rfc5702>>.
- [RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <<https://www.rfc-editor.org/info/rfc8032>>.

- [RFC8080] Sury, O. and R. Edmonds, "Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC", RFC 8080, DOI 10.17487/RFC8080, February 2017, <<https://www.rfc-editor.org/info/rfc8080>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.

8.2. Informative References

- [RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <<https://www.rfc-editor.org/info/rfc6376>>.
- [RFC7344] Kumari, W., Gudmundsson, O., and G. Barwood, "Automating DNSSEC Delegation Trust Maintenance", RFC 7344, DOI 10.17487/RFC7344, September 2014, <<https://www.rfc-editor.org/info/rfc7344>>.

Appendix A. Implementation (and Toy Deployment:-) Status

[[Note to RFC-editor: according to RFC 7942, sections such as this one ought not be part of the final RFC. We still dislike that idea, but whatever;-)]]

We are not aware of any independent implementations so far. One of the authors has a github repo at <<https://github.com/sftcd/rdbd-deebeedeerrr>> with scripts that allow one to produce zone file fragments and signatures for a set of domains. There is also a wrapper script for the dig tool that provides a nicer view of RDBD and RDBDKEY records, and that verifies signatures. See the README there for details.

In terms of deployments, we used the above for a "toy" deployment in the tolerantnetworks.ie domain and other related domains that one can determine by following the relevant trail:-)

Appendix B. Examples

These examples have been generated using the proof-of-concept implementation mentioned above. These are intended for interop, not for beauty:-) The dig wrapper script referred to above produces more readable output, shown further below..

The following names and other values are used in these examples.

- o Relating domain: my.example
- o Related domain: my-way.example
- o Unrelated domain: my-bad.example
- o URL for other related domains: <https://my.example/related-names>
- o URL for other unrelaed domains: <https://my.example/unrelateds>

my.example zone file fragments:

```

my.example.      3600 IN TYPE65448 \# 298 (
0000030830820122300d06092a864886f70d010101050
00382010f003082010a0282010100bb3b09979b3c4e61
0f231dafbd8295d5b6d9475eba8df1cfff49b08b99a768
15e660c243b8ce7175cc9857be00847cfff865ca81e56a
f0ec1813a43787902e8b2560b64016c4c8e64262b7b8e
ae2e6f735e1186237fff49110227b69fbcefa1cfddf7f
df052f250871bb03be114493a8e29a95d04b50b9e99b5
8e40e70381384c159d02d781e6837791c2ead0c547e7f
fb0aa198b2aef259c42273a69af4f22c7439972d3052d
4a581895e203115963689044b4cbbdb6cf90ff1866630
593aad625772e6f540bd93801c5781fdd74481fbb6399
f745b4525c767e3fb4a4d919e265d541f6bee95d0b9e1
15bd4749a3a9748e2d8745466629fa6682d36e83cbae8
30203010001
)
my.example.      3600 IN TYPE65443 \# 85 (
0001066d792d776179076578616d706c650039820f039
b08e9d5a8e057a87c6e7ddb92a680b7a2e69baef46404
b3bc9fcd93f4fe261bda56c107dba2d672255a86a771f
cc3eca0f12cdd1b302f20b2234de8610e03
)
my.example.      3600 IN TYPE65443 \# 18 (
0000066d792d626164076578616d706c6500
)
my.example.      3600 IN TYPE65443 \# 39 (
00012368747470733a2f2f6d792d7761792e6578616d7
06c652f6d7973747566662e6a736f6e00
)
my.example.      3600 IN TYPE65443 \# 42 (
00002668747470733a2f2f6d792d7761792e6578616d7
06c652f6e6f746d7973747566662e6a736f6e00
)

```


my.example private key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAuzsJl5s8TmEPIx2vvYKV1bbZR166jfHP9JsIuZp2gV5mDCQ7
jOcXXMmFe+AIR8/4ZcqB5Wrw7BgTpDeHkC6LJWC2QBbEyOZCYre46uLm9zXhGGI3
//SRECJ7afvO+hz933/fBS8lCHG7A74RRJOo4pqV0EtQuembWOQOcDgThMFZ0C14
Hmg3eRwurQxUfn/7CqGYsq7yWcQic6aa9PIsdDmXLtBS1KWBiv4gMRWWNokES0y7
22z5D/GGZjBZOqliv3Lm9UC9k4AcV4H910SB+7Y5n3RbRSXHZ+P7Sk2RniZdVB9r
7pXQueEVvUdJo6l0ji2HRUZmKfpmgtNug8uugwIDAQABAoIBAF1sJuwkBGJjocb2
4CLijtsVorMu/E0pdIdr+F2MSkdhD/BM//3drVWaJGXcMqWKizpXYptT0iUsG1jd
cGIsJzgeWrH96nEIG+XgIH/rei2uD8Q39hNcOCnh2szWXb+FSdQEnQacMJFFXfmbW
pw0dlK5FTi2h9wTdIKupF988y9h4OzVkw9qIDqOzKAKnxoyYZ0xiglaUq6NeHRs2
Sv7ow5CErKm4ZDqvtcqxS+uWblm3i5LsPGKexDZfXDQqle7hjFbXKUw+ZREF8hzc
bCfa3A5Xyo7nLdgGR2DOZlzoQA+iz5Cnbp35gdOV+giptlwndrn8Lc8U1Zf1f47T
aOxh2YECgYEA4u/VQ2B4Ux4NNX8g3womc/rJZOMWVxkd8odRhBy4s0c+atGy3ztp
SOPrBQrkjcFE831b596MOE11y1GpmKK7q5nI2IcMuStnLoj27a95QVznswnbyA6a
g3cIAz/loHCexLzi8edjcwTxJv1XNE9518SbkU0EbW2OY5jZsHU4I0MCgYEA0zVt
m3PrU5/JW1GqmRhDa7PyfB9ESq5mIXIaT6mPh0XLryMn2uUmFBMC3iuxNayjQgzI
Gg3XVC1cb4vvrvDrkxY5aTdmizvVf0MletBiLYjCwWHsOGql4hxwhvENYcYvCjs
T0WShG8FuuuHaH371+2hBkREeLHQRLyh0om2c8ECgYEA4JCb5PSNnRjB19hZWtzc
eGBu8lqVPNMqA1lMnQMe8qlJZsLj0mskIHd4N6Ez0eKyrJAcZjKfZwefzPaecOB3
/bNMQJhDSulcTXxTfZjq0HdzAIR87FcnJ3iegTi1R0iKk/ymRuLGUodNa1u+85DB
7XYsy3f/LZoAESasJCWay6kCgYAYGpuc5BvwY5iF5FK/LMVZuH+OuHAF801hI8tg
GI5m/cS7EHD0+aVV38ivYdgRLpowIg4aOCxb19AI2j6KdAbegsgpzyLx5sjmfYBt
1DhgsSyRacFvY0MH3aN289VRCXJxuJeOmqeOaTQHyrX9sN1ctQ+dB/biVvRcrL7q
ziaNQKBgQC9MECoVH/bYJVY6RoC5ZYAa6A4CYDhaXnw40lQ90ckSgWr3FenV7gw
b2xg7zLOX2HZZ+6HejMNGC/efZKVN2Okkpe4KGOXcDH3pYrrkLsLCNRXzxBSyOIt
e3elkAriqiXcr3sPBbn7nakUa7G23O7Hb31C0KGM9f9znN+qWda+3g==
-----END RSA PRIVATE KEY-----
```

my-way.example zone file fragments:

```

my-way.example. 3600 IN TYPE65448 \# 36 (
    0000030f6d5a2d3caf0d740e139d36a0e52325c4e078e
    7623f19be3b872367dc8027ef42
)
my-way.example. 3600 IN TYPE65443 \# 273 (
    0001026d79076578616d706c65003e6c088d887950e26
    305a59bbe63263b65d34e11656968497500cbef7af12b
    e14d173d7368e24da54258c851456d3c2d94437692879
    d1d2b5d3f0acf1e3de6ebb345f8c31f209af6fd7f2731
    3804fc79db421231126e3e42115ce51a81d2619ed221a
    fea2b64d1d9ffbef0bd4786fbe5f42c75951ae645078d
    b7a5a88ed3173d4a209734f49a23a0920ce38ed44011d
    784e47cf7658cc313cf01349c80b936b17fca3542f32a
    ff956e808c2520736a917df648e4e5f2eea5de994ce90
    dba6d5051a4e0934da4a9f6ff01ef5df98d3b4da52b12
    ea3b8e7ebabcf6d7a0a170dc1284753e3e6b039f8a32c
    e707312ea5b02180072b517a6056db6e47f8dd5240ab1
    874646
)

```

my-way.example private key:

```

0000000 5f24 3132 daa0 4cc4 0a77 4cb6 e834 16db
0000020 05b0 faf7 ca27 16b6 0ae7 e177 d3f9 db5f
0000040

```

Appendix C. Possible dig output...

Below we show the output that a modified dig tool might display for the my.example assertions above.

```
$ dig RDBD my.example

; <<>> DiG 9.11.5-P1-lubuntu2.5-Ubuntu <<>> RDBD my.example
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4289
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: e69085d4b9a18cca63ae96035d7bc0aa96580e0d6255c122 (good)
;; QUESTION SECTION:
;my.example. IN RDBD

;; ANSWER SECTION:
my.example. 3600 IN RDBD RELATED may-way.example Sig: good
                KeyId: 50885 Alg: 15 Sig: UIi04agb...
my.example. 3600 IN RDBD UNRELATED my-bad.example
my.example. 3600 IN RDBD RELATED https://my-way.example/mystuff.json
my.example. 3600 IN RDBD UNRELATED https://my-way.example/notmine.json

;; Query time: 721 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Sep 13 17:15:38 IST 2019
;; MSG SIZE rcvd: 600
```

Appendix D. Changes and Open Issues

[[RFC editor: please delete this appendix]]

D.1. Changes from -02 to -03

- o Incorporated feedback/comments from IETF-105
- o Suggest list dicussion move to dnsop@ietf.org
- o Adopted some experimental RRCODE values
- o Fixed normative vs. informative refs
- o Changed the examples to use the PoC implementation.
- o Restructured text a lot

D.2. Changes from -01 to -02

- o Added negative assertions based on IETF104 feedback
- o Added URL option based on IETF104 feedback
- o Made sample generation script
- o Typo fixes etc.

D.3. Changes from -00 to -01

- o Changed from primary/secondary to relating/related (better suggestions are still welcome)
- o Moved away from abuse of TXT RRs
- o We now specify optional DNSSEC-like signatures (we'd be fine with moving back to a more DKIM-like mechanism, but wanted to see how this looked)
- o Added Ed25519 option
- o Re-worked and extended examples

Authors' Addresses

Alex Brotman
Comcast, Inc

Email: alex_brotman@comcast.com

Stephen Farrell
Trinity College Dublin

Email: stephen.farrell@cs.tcd.ie

Domain Name System Operations
Internet-Draft
Updates: 1123, 1536 (if approved)
Intended status: Best Current Practice
Expires: 10 July 2022

J.T. Kristoff
DataPlane.org
D. Wessels
Verisign
6 January 2022

DNS Transport over TCP - Operational Requirements
draft-ietf-dnsop-dns-tcp-requirements-15

Abstract

This document updates RFC 1123 and RFC 1536. This document requires the operational practice of permitting DNS messages to be carried over TCP on the Internet as a Best Current Practice. This operational requirement is aligned with the implementation requirements in RFC 7766. The use of TCP includes both DNS over unencrypted TCP, as well as over an encrypted TLS session. The document also considers the consequences of this form of DNS communication and the potential operational issues that can arise when this Best Current Practice is not upheld.

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 10 July 2022.

Copyright Notice

Copyright (c) 2022 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 (<https://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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	4
1.1. Requirements Language	4
2. History of DNS over TCP	4
2.1. Uneven Transport Usage and Preference	5
2.2. Waiting for Large Messages and Reliability	5
2.3. EDNS(0)	6
2.4. Fragmentation and Truncation	6
2.5. "Only Zone Transfers Use TCP"	8
2.6. Reuse, Pipelining, and Out-of-Order Processing	8
3. DNS over TCP Requirements	9
4. Network and System Considerations	10
4.1. Connection Establishment and Admission	10
4.2. Connection Management	12
4.3. Connection Termination	13
4.4. DNS-over-TLS	13
4.5. Defaults and Recommended Limits	14
5. DNS over TCP Filtering Risks	15
5.1. Truncation, Retries, and Timeouts	15
5.2. DNS Root Zone KSK Rollover	16
6. Logging and Monitoring	16
7. IANA Considerations	17
8. Security Considerations	17
9. Privacy Considerations	18
10. Acknowledgments	18
11. References	18
11.1. Normative References	18
11.2. Informative References	19
Appendix A. Standards Related to DNS Transport over TCP	27
A.1. IETF RFC 1035 - DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION	27
A.2. IETF RFC 1536 - Common DNS Implementation Errors and Suggested Fixes	27
A.3. IETF RFC 1995 - Incremental Zone Transfer in DNS	27
A.4. IETF RFC 1996 - A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)	27
A.5. IETF RFC 2181 - Clarifications to the DNS Specification	27
A.6. IETF RFC 2694 - DNS extensions to Network Address Translators (DNS_ALG)	28
A.7. IETF RFC 3225 - Indicating Resolver Support of DNSSEC	28

A.8.	IETF RFC 3226 - DNSSEC and IPv6 A6 aware server/resolver message size requirements	28
A.9.	IETF RFC 4472 - Operational Considerations and Issues with IPv6 DNS	28
A.10.	IETF RFC 5452 - Measures for Making DNS More Resilient against Forged Answers	28
A.11.	IETF RFC 5507 - Design Choices When Expanding the DNS . .	29
A.12.	IETF RFC 5625 - DNS Proxy Implementation Guidelines . . .	29
A.13.	IETF RFC 5936 - DNS Zone Transfer Protocol (AXFR)	29
A.14.	IETF RFC 7534 - AS112 Nameserver Operations	29
A.15.	IETF RFC 6762 - Multicast DNS	29
A.16.	IETF RFC 6891 - Extension Mechanisms for DNS (EDNS(0)) .	29
A.17.	IETF RFC 6950 - Architectural Considerations on Application Features in the DNS	30
A.18.	IETF RFC 7477 - Child-to-Parent Synchronization in DNS .	30
A.19.	IETF RFC 7720 - DNS Root Name Service Protocol and Deployment Requirements	30
A.20.	IETF RFC 7766 - DNS Transport over TCP - Implementation Requirements	30
A.21.	IETF RFC 7828 - The edns-tcp-keepalive EDNS(0) Option . .	30
A.22.	IETF RFC 7858 - Specification for DNS over Transport Layer Security (TLS)	31
A.23.	IETF RFC 7873 - Domain Name System (DNS) Cookies	31
A.24.	IETF RFC 7901 - CHAIN Query Requests in DNS	31
A.25.	IETF RFC 8027 - DNSSEC Roadblock Avoidance	31
A.26.	IETF RFC 8094 - DNS over Datagram Transport Layer Security (DTLS)	32
A.27.	IETF RFC 8162 - Using Secure DNS to Associate Certificates with Domain Names for S/MIME	32
A.28.	IETF RFC 8324 - DNS Privacy, Authorization, Special Uses, Encoding, Characters, Matching, and Root Structure: Time for Another Look?	32
A.29.	IETF RFC 8467 - Padding Policies for Extension Mechanisms for DNS (EDNS(0))	32
A.30.	IETF RFC 8482 - Providing Minimal-Sized Responses to DNS Queries That Have QTYPE=ANY	32
A.31.	IETF RFC 8483 - Yeti DNS Testbed	33
A.32.	IETF RFC 8484 - DNS Queries over HTTPS (DoH)	33
A.33.	IETF RFC 8490 - DNS Stateful Operations	33
A.34.	IETF RFC 8501 - Reverse DNS in IPv6 for Internet Service Providers	33
A.35.	IETF RFC 8806 - Running a Root Server Local to a Resolver	33
A.36.	IETF RFC 8906 - A Common Operational Problem in DNS Servers: Failure to Communicate	33
A.37.	IETF RFC 8932 - Recommendations for DNS Privacy Service Operators	34

A.38. IETF RFC 8945 - Secret Key Transaction Authentication for	
DNS (TSIG)	34
Authors' Addresses	34

1. Introduction

DNS messages are delivered using UDP or TCP communications. While most DNS transactions are carried over UDP, some operators have been led to believe that any DNS over TCP traffic is unwanted or unnecessary for general DNS operation. When DNS over TCP has been restricted, a variety of communication failures and debugging challenges often arise. As DNS and new naming system features have evolved, TCP as a transport has become increasingly important for the correct and safe operation of an Internet DNS. Reflecting modern usage, the DNS standards declare that support for TCP is a required part of the DNS implementation specifications [RFC7766]. This document is the formal requirements equivalent for the operational community, encouraging system administrators, network engineers, and security staff to ensure DNS over TCP communications support is on par with DNS over UDP communications. It updates [RFC1123] Section 6.1.3.2 to clarify that all DNS resolvers and recursive servers MUST support and service both TCP and UDP queries, and also updates [RFC1536] to remove the misconception that TCP is only useful for zone transfers.

1.1. Requirements Language

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. History of DNS over TCP

The curious state of disagreement between operational best practices and guidance for DNS transport protocols derives from conflicting messages operators have received from other operators, implementors, and even the IETF. Sometimes these mixed signals have been explicit; on other occasions, conflicting messages have been implicit. This section presents an interpretation of the storied and conflicting history that led to this document. This section is included for informational purposes only.

2.1. Uneven Transport Usage and Preference

In the original suite of DNS specifications, [RFC1034] and [RFC1035] clearly specified that DNS messages could be carried in either UDP or TCP, but they also stated a preference for UDP as the best transport for queries in the general case. As stated in [RFC1035]:

"While virtual circuits can be used for any DNS activity, datagrams are preferred for queries due to their lower overhead and better performance."

Another early, important, and influential document, [RFC1123], marked the preference for a transport protocol more explicitly:

"DNS resolvers and recursive servers MUST support UDP, and SHOULD support TCP, for sending (non-zone-transfer) queries."

and further stipulated:

"A name server MAY limit the resources it devotes to TCP queries, but it SHOULD NOT refuse to service a TCP query just because it would have succeeded with UDP."

Culminating in [RFC1536], DNS over TCP came to be associated primarily with the zone transfer mechanism, while most DNS queries and responses were seen as the dominion of UDP.

2.2. Waiting for Large Messages and Reliability

In the original specifications, the maximum DNS over UDP message size was enshrined at 512 bytes. However, even while [RFC1123] preferred UDP for non-zone transfer queries, it foresaw DNS over TCP becoming more popular in the future to overcome this limitation:

"[...] it is also clear that some new DNS record types defined in the future will contain information exceeding the 512 byte limit that applies to UDP, and hence will require TCP."

At least two new, widely anticipated developments were set to elevate the need for DNS over TCP transactions. The first was dynamic updates defined in [RFC2136] and the second was the set of extensions collectively known as DNSSEC, whose operational considerations are originally given in [RFC2541]. The former suggested "requestors who require an accurate response code must use TCP," while the latter warned "... larger keys increase the size of KEY and SIG RRs. This increases the chance of DNS UDP packet overflow and the possible necessity for using higher overhead TCP in responses."

Yet, defying some expectations, DNS over TCP remained little-used in real traffic across the Internet in the late 1990s. Dynamic updates saw little deployment between autonomous networks. Around the time DNSSEC was first defined, another new feature helped solidify UDP transport dominance for message transactions.

2.3. EDNS(0)

In 1999 the IETF published the Extension Mechanisms for DNS (EDNS(0)) in [RFC2671] (superseded in 2013 by an update in [RFC6891]). That document standardized a way for communicating DNS nodes to perform rudimentary capabilities negotiation. One such capability written into the base specification and present in every EDNS(0)-compatible message is the value of the maximum UDP payload size the sender can support. This unsigned 16-bit field specifies, in bytes, the maximum (possibly fragmented) DNS message size a node is capable of receiving over UDP. In practice, typical values are a subset of the 512- to 4096-byte range. EDNS(0) became widely deployed over the next several years, and numerous surveys ([CASTRO2010], [NETALYZR]) have shown that many systems support larger UDP MTUs with EDNS(0).

The natural effect of EDNS(0) deployment meant DNS messages larger than 512 bytes would be less reliant on TCP than they might otherwise have been. While a non-negligible population of DNS systems lacked EDNS(0) or fell back to TCP when necessary, DNS clients still strongly prefer UDP to TCP. For example, as of 2014, DNS over TCP transactions remained a very small fraction of overall DNS traffic received by root name servers [VERISIGN].

2.4. Fragmentation and Truncation

Although EDNS(0) provides a way for endpoints to signal support for DNS messages exceeding 512 bytes, the realities of a diverse and inconsistently deployed Internet may result in some large messages being unable to reach their destination. Any IP datagram whose size exceeds the MTU of a link it transits will be fragmented and then reassembled by the receiving host. Unfortunately, it is not uncommon for middleboxes and firewalls to block IP fragments. If one or more fragments do not arrive, the application does not receive the message and the request times out.

For IPv4-connected hosts, the MTU is often the Ethernet payload size of 1500 bytes. This means that the largest unfragmented UDP DNS message that can be sent over IPv4 is likely 1472 bytes, although tunnel encapsulation may reduce that maximum message size in some cases.

For IPv6, the situation is a little more complicated. First, IPv6 headers are 40 bytes (versus 20 without options in IPv4). Second, approximately one third of DNS recursive resolvers use the minimum MTU of 1280 bytes [APNIC]. Third, fragmentation in IPv6 can only be done by the host originating the datagram. The need to fragment is conveyed in an ICMPv6 "packet too big" message. The originating host indicates a fragmented datagram with IPv6 extension headers. Unfortunately, it is quite common for both ICMPv6 and IPv6 extension headers to be blocked by middleboxes. According to [HUSTON] some 35% of IPv6-capable recursive resolvers were unable to receive a fragmented IPv6 packet. When the originating host receives a signal that fragmentation is required, it is expected to populate its Path MTU cache for that destination. The application, then, will retry the query after a timeout since the host does not generally retain copies of messages sent over UDP for potential retransmission.

The practical consequence of all this is that DNS requestors must be prepared to retry queries with different EDNS(0) maximum message size values. Administrators of [BIND] are likely to be familiar with seeing "success resolving ... after reducing the advertised EDNS(0) UDP packet size to 512 octets" messages in their system logs.

Often, reducing the EDNS(0) UDP packet size leads to a successful response. That is, the necessary data fits within the smaller message size. However, when the data does not fit, the server sets the truncated flag in its response, indicating the client should retry over TCP to receive the whole response. This is undesirable from the client's point of view because it adds more latency and potentially undesirable from the server's point of view due to the increased resource requirements of TCP.

Note that a receiver is unable to differentiate between packets lost due to congestion and packets (fragments) intentionally dropped by firewalls or middleboxes. Over network paths with non-trivial amounts of packet loss, larger, fragmented DNS responses are more likely to never arrive and time out compared to smaller, unfragmented responses. Clients might be misled into retrying queries with different EDNS(0) UDP packet size values for the wrong reason.

The issues around fragmentation, truncation, and TCP are driving certain implementation and policy decisions in the DNS. Notably, Cloudflare implemented what it calls "DNSSEC black lies" [CLOUDFLARE] and uses ECDSA algorithms, such that their signed responses fit easily in 512 bytes. The Key Signing Key (KSK) Rollover design team [DESIGNTEAM] spent a lot of time thinking and worrying about response sizes. There is growing sentiment in the DNSSEC community that RSA key sizes beyond 2048-bits are impractical and that critical infrastructure zones should transition to elliptic curve algorithms to keep response sizes manageable [ECDSA].

More recently, renewed security concerns about fragmented DNS messages ([AVOID_FRAGS], [FRAG_POISON]) are leading implementors to consider smaller responses and lower default EDNS(0) UDP payload size values for both queriers and responders [FLAGDAY2020].

2.5. "Only Zone Transfers Use TCP"

Today, the majority of the DNS community expects, or at least has a desire, to see DNS over TCP transactions occur without interference [FLAGDAY2020]. However, there has also been a long-held belief by some operators, particularly for security-related reasons, that DNS over TCP services should be purposely limited or not provided at all [CHES94], [DJBDNS]. A popular meme is that DNS over TCP is only ever used for zone transfers and is generally unnecessary otherwise, with filtering all DNS over TCP traffic even described as a best practice.

The position on restricting DNS over TCP had some justification given that historical implementations of DNS nameservers provided very little in the way of TCP connection management (for example see Section 6.1.2 of [RFC7766] for more details). However, modern standards and implementations are nearing parity with the more sophisticated TCP management techniques employed by, for example, HTTP(S) servers and load balancers.

2.6. Reuse, Pipelining, and Out-of-Order Processing

The idea that a TCP connection can support multiple transactions goes back as far as [RFC0883], which states: "Multiple messages may be sent over a virtual circuit." Although [RFC1035], which updates the former, omits this particular detail, it has been generally accepted that a TCP connection can be used for more than one query and response.

[RFC5966] clarified that servers are not required to preserve the order of queries and responses over any transport. [RFC7766], which updates the former, further encourages query pipelining over TCP to achieve performance on par with UDP. A server that sends out-of-

order responses to pipelined queries avoids head-of-line blocking when the response for a later query is ready before the response to an earlier query.

However, TCP can potentially suffer from a different head-of-line blocking problem due to packet loss. Since TCP itself enforces ordering, a single lost segment delays delivery of data in any following segments until the lost segment is retransmitted and successfully received.

3. DNS over TCP Requirements

An average increase in DNS message size (e.g., due to DNSSEC), the continued development of new DNS features (Appendix A), and a denial of service mitigation technique (Section 8), all show that DNS over TCP transactions are as important to the correct and safe operation of the Internet DNS as ever, if not more so. Furthermore, there has been research that argues connection-oriented DNS transactions may provide security and privacy advantages over UDP transport [TDNS]. In fact, the standard for DNS over TLS [RFC7858] is just this sort of specification. Therefore, this document makes explicit that it is undesirable for network operators to artificially inhibit DNS over TCP transport.

Section 6.1.3.2 in [RFC1123] is updated: All DNS resolvers and servers MUST support and service both UDP and TCP queries.

- * DNS servers (including forwarders) MUST support and service TCP for receiving queries, so that clients can reliably receive responses that are larger than what either side considers too large for UDP.
- * DNS clients MUST support TCP for sending queries, so that they can retry truncated UDP responses as necessary.

Furthermore, the requirement in Section 6.1.3.2 of [RFC1123] around limiting the resources a server devotes to queries is hereby updated:

OLD:

A name server MAY limit the resources it devotes to TCP queries, but it SHOULD NOT refuse to service a TCP query just because it would have succeeded with UDP.

NEW:

A name server MAY limit the resources it devotes to queries, but it MUST NOT refuse to service a query just because it would have succeeded with another transport protocol.

Lastly, Section 1 of [RFC1536] is updated to eliminate the misconception that TCP is only useful for zone transfers:

OLD:

DNS implements the classic request-response scheme of client-server interaction. UDP is, therefore, the chosen protocol for communication though TCP is used for zone transfers.

NEW:

DNS implements the classic request-response scheme of client-server interaction.

Filtering of DNS over TCP is harmful in the general case. DNS resolver and server operators MUST support and provide DNS service over both UDP and TCP transports. Likewise, network operators MUST allow DNS service over both UDP and TCP transports. It is acknowledged that DNS over TCP service can pose operational challenges that are not present when running DNS over UDP alone, and vice-versa. However, the potential damage incurred by prohibiting DNS over TCP service is more detrimental to the continued utility and success of the DNS than when its usage is allowed.

4. Network and System Considerations

This section describes measures that systems and applications can take to optimize performance over TCP and to protect themselves from TCP-based resource exhaustion and attacks.

4.1. Connection Establishment and Admission

Resolvers and other DNS clients should be aware that some servers might not be reachable over TCP. For this reason, clients MAY track and limit the number of TCP connections and connection attempts to a single server. Reachability problems can be caused by network elements close to the server, close to the client, or anywhere along the path between them. Mobile clients that cache connection failures MAY do so on a per-network basis, or MAY clear such a cache upon change of network.

Additionally, DNS clients MAY enforce a short timeout on unestablished connections, rather than rely on the host operating system's TCP connection timeout, which is often around 60-120 seconds

(i.e., due to an initial retransmission timeout of 1 second, the exponential back off rules of [RFC6298], and a limit of six retries as is the default in Linux).

The SYN flooding attack is a denial-of-service method affecting hosts that run TCP server processes [RFC4987]. This attack can be very effective if not mitigated. One of the most effective mitigation techniques is SYN cookies, described in Section 3.6 of [RFC4987], which allows the server to avoid allocating any state until the successful completion of the three-way handshake.

Services not intended for use by the public Internet, such as most recursive name servers, SHOULD be protected with access controls. Ideally these controls are placed in the network, well before any unwanted TCP packets can reach the DNS server host or application. If this is not possible, the controls can be placed in the application itself. In some situations (e.g. attacks) it may be necessary to deploy access controls for DNS services that should otherwise be globally reachable. See also [RFC5358].

The FreeBSD and NetBSD operating systems have an "accept filter" feature ([accept_filter]) that postpones delivery of TCP connections to applications until a complete, valid request has been received. The `dns_accf(9)` filter ensures that a valid DNS message is received. If not, the bogus connection never reaches the application. The Linux `TCP_DEFER_ACCEPT` feature, while more limited in scope, can provide some of the same benefits as the BSD accept filter feature. These features are implemented as low-level socket options, and are not activated automatically. If applications wish to use these features, they need to make specific calls to set the right options, and administrators may also need to configure the applications to appropriately use the features.

Per [RFC7766], applications and administrators are advised to remember that TCP MAY be used before sending any UDP queries. Networks and applications MUST NOT be configured to refuse TCP queries that were not preceded by a UDP query.

TCP Fast Open [RFC7413] (TFO) allows TCP clients to shorten the handshake for subsequent connections to the same server. TFO saves one round-trip time in the connection setup. DNS servers SHOULD enable TFO when possible. Furthermore, DNS servers clustered behind a single service address (e.g., anycast or load-balancing), SHOULD either use the same TFO server key on all instances, or disable TFO for all members of the cluster.

DNS clients MAY also enable TFO. At the time of this writing, on some operating systems it is not implemented, or is disabled by default. [WIKIPEDIA_TFO] describes applications and operating systems that support TFO.

4.2. Connection Management

Since host memory for TCP state is a finite resource, DNS clients and servers SHOULD actively manage their connections. Applications that do not actively manage their connections can encounter resource exhaustion leading to denial of service. For DNS, as in other protocols, there is a tradeoff between keeping connections open for potential future use and the need to free up resources for new connections that will arrive.

Operators of DNS server software SHOULD be aware that operating system and application vendors MAY impose a limit on the total number of established connections. These limits may be designed to protect against DDoS attacks or performance degradation. Operators SHOULD understand how to increase these limits if necessary, and the consequences of doing so. Limits imposed by the application SHOULD be lower than limits imposed by the operating system, so that the application can apply its own policy to connection management, such as closing the oldest idle connections first.

DNS server software MAY provide a configurable limit on the number of established connections per source IP address or subnet. This can be used to ensure that a single or small set of users cannot consume all TCP resources and deny service to other users. Note, however, that if this limit is enabled, it possibly limits client performance while leaving some TCP resources unutilized. Operators SHOULD be aware of these tradeoffs and ensure this limit, if configured, is set appropriately based on the number and diversity of their users, and whether users connect from unique IP addresses or through a shared Network Address Translator [RFC3022].

DNS server software SHOULD provide a configurable timeout for idle TCP connections. This can be used to free up resources for new connections and to ensure that idle connections are eventually closed. At the same time, it possibly limits client performance while leaving some TCP resources unutilized. For very busy name servers this might be set to a low value, such as a few seconds. For less busy servers it might be set to a higher value, such as tens of seconds. DNS clients and servers SHOULD signal their timeout values using the edns-tcp-keepalive option [RFC7828].

DNS server software MAY provide a configurable limit on the number of transactions per TCP connection. This can help protect against unfair connection use (e.g., not releasing connection slots to other clients) and network evasion attacks.

Similarly, DNS server software MAY provide a configurable limit on the total duration of a TCP connection. This can help protect against unfair connection use, slow read attacks, and network evasion attacks.

Since clients may not be aware of server-imposed limits, clients utilizing TCP for DNS need to always be prepared to re-establish connections or otherwise retry outstanding queries.

4.3. Connection Termination

The TCP peer that initiates a connection close retains the socket in the TIME_WAIT state for some amount of time, possibly a few minutes. It is generally preferable for clients to initiate the close of a TCP connection so that busy servers do not accumulate many sockets in the TIME_WAIT state, which can cause performance problems or even denial of service. The edns-tcp-keepalive EDNS(0) option [RFC7828] can be used to encourage clients to close connections.

On systems where large numbers of sockets in TIME_WAIT are observed (either as client or server), and are affecting an application's performance, it may be tempting to tune local TCP parameters. For example, the Linux kernel has a "sysctl" parameter named net.ipv4.tcp_tw_reuse which allows connections in the TIME_WAIT state to be reused in specific circumstances. Note, however, this affects only outgoing (client) connections and has no impact on servers. In most cases it is NOT RECOMMENDED to change parameters related to the TIME_WAIT state. It should only be done by those with detailed knowledge of both TCP and the affected application.

4.4. DNS-over-TLS

DNS messages may be sent over TLS to provide privacy between stubs and recursive resolvers. [RFC7858] is a Standards Track document describing how this works. Although DNS-over-TLS utilizes TCP port 853 instead of port 53, this document applies equally well to DNS-over-TLS. Note, however, DNS-over-TLS is only defined between stubs and recursives at the time of this writing.

The use of TLS places even stronger operational burdens on DNS clients and servers. Cryptographic functions for authentication and encryption require additional processing. Unoptimized connection setup with TLS 1.3 [RFC8446] takes one additional round-trip compared

to TCP. Connection setup times can be reduced with TCP Fast Open, and TLS False Start [RFC7918] for TLS 1.2. TLS 1.3 session resumption does not reduce round-trip latency because no application profile for use of TLS 0-RTT data with DNS has been published at the time of this writing. However, TLS session resumption can reduce the number of cryptographic operations, and in TLS 1.2, session resumption does reduce the number of additional round trips from two to one.

4.5. Defaults and Recommended Limits

A survey of features and defaults was conducted for popular open source DNS server implementations at the time of writing. This section documents those defaults and makes recommendations for configurable limits that can be used in the absence of any other information. Any recommended values in this document are only intended as a starting point for administrators that are unsure what sorts of limits might be reasonable. Operators SHOULD use application-specific monitoring, system logs, and system monitoring tools to gauge whether their service is operating within or exceeding these limits, and adjust accordingly.

Most open source DNS server implementations provide a configurable limit on the total number of established connections. Default values range from 20 to 150. In most cases, where the majority of queries take place over UDP, 150 is a reasonable limit. For services or environments where most queries take place over TCP or TLS, 5000 is a more appropriate limit.

Only some open source implementations provide a way to limit the number of connections per source IP address or subnet, but the default is to have no limit. For environments or situations where it may be necessary to enable this limit, 25 connections per source IP address is a reasonable starting point. The limit should be increased when aggregated by subnet, or for services where most queries take place over TCP or TLS.

Most open source implementations provide a configurable idle timeout on connections. Default values range from 2 to 30 seconds. In most cases, 10 seconds is a reasonable default for this limit. Longer timeouts improve connection reuse, but busy servers may need to use a lower limit.

Only some open source implementations provide a way to limit the number of transactions per connection, but the default is to have no limit. This document does not offer advice on particular values for such a limit.

Only some open source implementations provide a way to limit the duration of connection, but the default is to have no limit. This document does not offer advice on particular values for such a limit.

5. DNS over TCP Filtering Risks

Networks that filter DNS over TCP risk losing access to significant or important pieces of the DNS namespace. For a variety of reasons a DNS answer may require a DNS over TCP query. This may include large message sizes, lack of EDNS(0) support, DDoS mitigation techniques (including [RRL]), or perhaps some future capability that is as yet unforeseen will also demand TCP transport.

For example, [RFC7901] describes a latency-avoiding technique that sends extra data in DNS responses. This makes responses larger and potentially increases the effectiveness of DDoS reflection attacks. The specification mandates the use of TCP or DNS Cookies [RFC7873].

Even if any or all particular answers have consistently been returned successfully with UDP in the past, this continued behavior cannot be guaranteed when DNS messages are exchanged between autonomous systems. Therefore, filtering of DNS over TCP is considered harmful and contrary to the safe and successful operation of the Internet. This section enumerates some of the known risks at the time of this writing when networks filter DNS over TCP.

5.1. Truncation, Retries, and Timeouts

Networks that filter DNS over TCP may inadvertently cause problems for third-party resolvers as experienced by [TOYAMA]. For example, a resolver receives queries for a moderately popular domain. The resolver forwards the queries to the domain's authoritative name servers, but those servers respond with the TC bit set. The resolver retries over TCP, but the authoritative server blocks DNS over TCP. The pending connections consume resources on the resolver until they time out. If the number and frequency of these truncated-and-then-blocked queries is sufficiently high, the resolver wastes valuable resources on queries that can never be answered. This condition is generally not easily or completely mitigated by the affected DNS resolver operator.

5.2. DNS Root Zone KSK Rollover

The plans for deploying a new root zone DNSSEC KSK highlighted a potential problem in retrieving the root zone key set [LEWIS]. During some phases of the KSK rollover process, root zone DNSKEY responses were larger than 1280 bytes, the IPv6 minimum MTU for links carrying IPv6 traffic [RFC8200]. There was some concern [KSK_ROLLOVER_ARCHIVES] that any DNS server unable to receive large DNS messages over UDP, or any DNS message over TCP, would experience disruption while performing DNSSEC validation.

However, during the year-long postponement of the KSK rollover there were no reported problems that could be attributed to the 1414 octet DNSKEY response when both the old and new keys were published in the zone. Additionally, there were no reported problems during the two-month period when the old key was published as revoked and the DNSKEY response was 1425 octets in size [ROLL_YOUR_ROOT].

6. Logging and Monitoring

Developers of applications that log or monitor DNS SHOULD NOT ignore TCP due to the perception that it is rarely used or is hard to process. Operators SHOULD ensure that their monitoring and logging applications properly capture DNS messages over TCP. Otherwise, attacks, exfiltration attempts, and normal traffic may go undetected.

DNS messages over TCP are in no way guaranteed to arrive in single segments. In fact, a clever attacker might attempt to hide certain messages by forcing them over very small TCP segments. Applications that capture network packets (e.g., with libpcap [libpcap]) SHOULD implement and perform full TCP stream reassembly and analyze the reassembled stream instead of the individual packets. Otherwise, they are vulnerable to network evasion attacks [phrack]. Furthermore, such applications need to protect themselves from resource exhaustion attacks by limiting the amount of memory allocated to tracking unacknowledged connection state data. dnscap [dnscap] is an open-source example of a DNS logging program that implements TCP stream reassembly.

Developers SHOULD also keep in mind connection reuse, query pipelining, and out-of-order responses when building and testing DNS monitoring applications.

As an alternative to packet capture, some DNS server software supports dnstap [dnstap] as an integrated monitoring protocol intended to facilitate wide-scale DNS monitoring.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

This document, providing operational requirements, is the companion to the implementation requirements of DNS over TCP, provided in [RFC7766]. The security considerations from [RFC7766] still apply.

Ironically, returning truncated DNS over UDP answers in order to induce a client query to switch to DNS over TCP has become a common response to source address spoofed, DNS denial-of-service attacks [RRL]. Historically, operators have been wary of TCP-based attacks, but in recent years, UDP-based flooding attacks have proven to be the most common protocol attack on the DNS. Nevertheless, a high rate of short-lived DNS transactions over TCP may pose challenges. In fact, [DAI21] details a class of IP fragmentation attacks on DNS transactions if the IP Identifier field (16 bits in IPv4 and 32 bits in IPv6) can be predicted and a system is coerced to fragment rather than retransmit messages. While many operators have provided DNS over TCP service for many years without duress, past experience is no guarantee of future success.

DNS over TCP is similar to many other Internet TCP services. TCP threats and many mitigation strategies have been well-documented in a series of documents such as [RFC4953], [RFC4987], [RFC5927], and [RFC5961].

As mentioned in Section 6, applications that implement TCP stream reassembly need to limit the amount of memory allocated to connection tracking. A failure to do so could lead to a total failure of the logging or monitoring application. Imposition of resource limits creates a tradeoff between allowing some stream reassembly to continue and allowing some evasion attacks to succeed.

This document recommends that DNS Servers enable TFO when possible. [RFC7413] recommends that a pool of servers behind a load balancer with shared server IP address also share the key used to generate Fast Open cookies, to prevent inordinate fallback to the 3WHS. This guidance remains accurate, but comes with a caveat: compromise of one server would reveal this group-shared key, and allow for attacks involving the other servers in the pool by forging invalid Fast Open cookies.

9. Privacy Considerations

Since DNS over both UDP and TCP uses the same underlying message format, the use of one transport instead of the other does not change the privacy characteristics of the message content (i.e., the name being queried). A number of protocols have recently been developed to provide DNS privacy, including DNS over TLS [RFC7858], DNS over DTLS [RFC8094], DNS over HTTPS [RFC8484], with even more on the way.

Because TCP is somewhat more complex than UDP, some characteristics of a TCP conversation may enable DNS client fingerprinting and tracking that is not possible with UDP. For example, the choice of initial sequence numbers, window size, and options might be able to identify a particular TCP implementation, or even individual hosts behind shared resources such as network address translators (NATs).

10. Acknowledgments

This document was initially motivated by feedback from students who pointed out that they were hearing contradictory information about filtering DNS over TCP messages. Thanks in particular to a teaching colleague, JPL, who perhaps unknowingly encouraged the initial research into the differences between what the community has historically said and did. Thanks to all the NANOG 63 attendees who provided feedback to an early talk on this subject.

The following individuals provided an array of feedback to help improve this document: Joe Abley, Piet Barber, Sara Dickinson, Tony Finch, Bob Harold, Paul Hoffman, Geoff Huston, Tatuja Jinmei, Puneet Sood, and Richard Wilhelm. The authors are also indebted to the contributions stemming from discussion in the tcpm working group meeting at IETF 104. Any remaining errors or imperfections are the sole responsibility of the document authors.

11. References

11.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://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, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <<https://www.rfc-editor.org/info/rfc2181>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/info/rfc6891>>.
- [RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <<https://www.rfc-editor.org/info/rfc7766>>.
- [RFC7828] Wouters, P., Abley, J., Dickinson, S., and R. Bellis, "The edns-tcp-keepalive EDNS0 Option", RFC 7828, DOI 10.17487/RFC7828, April 2016, <<https://www.rfc-editor.org/info/rfc7828>>.
- [RFC7873] Eastlake 3rd, D. and M. Andrews, "Domain Name System (DNS) Cookies", RFC 7873, DOI 10.17487/RFC7873, May 2016, <<https://www.rfc-editor.org/info/rfc7873>>.
- [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/info/rfc8174>>.

11.2. Informative References

- [accept_filter] FreeBSD, "FreeBSD accept_filter(9)", 7 May 2018, <https://www.freebsd.org/cgi/man.cgi?query=accept_filter>.
- [APNIC] Huston, G., "DNS XL", October 2020, <<https://labs.apnic.net/?p=1380>>.
- [AVOID_FRAGS] Fujiwara, K. and P. Vixie, "Fragmentation Avoidance in DNS", Work in Progress, draft-ietf-dnsop-avoid-fragmentation-05, February 2021.
- [BIND] Internet Systems Consortium, "BIND 9 - ISC", April 2021, <<https://www.isc.org/bind/>>.
- [CASTRO2010] Castro, S., Zhang, M., John, W., Wessels, D., and k.c. claffy, "Understanding and preparing for DNS evolution", 2010.

- [CHES94] Cheswick, W.R. and S.M. Bellovin, "Firewalls and Internet Security: Repelling the Wily Hacker", 1994.
- [CLOUDFLARE]
Grant, D., "Economical With The Truth: Making DNSSEC Answers Cheap", 24 June 2016,
<<https://blog.cloudflare.com/black-lies/>>.
- [DAI21] Tianxiang, T., Shulman, H., and M. Waidner, "DNS-over-TCP Considered Vulnerable", 2021.
- [DESIGNTEAM]
Design Team Report, "Root Zone KSK Rollover Plan", 18 December 2015, <<https://www.iana.org/reports/2016/root-ksk-rollover-design-20160307.pdf>>.
- [DJBDNS] D.J. Bernstein, "When are TCP queries sent?", 2002,
<<https://cr.yp.to/djbdns/tcp.html#why>>.
- [dnscap] DNS-OARC, "DNSCAP", 7 May 2018,
<<https://www.dns-oarc.net/tools/dnscap>>.
- [dnstap] Edmonds, R. and P. Vixie, "dnstap", 7 May 2018,
<<https://dnstap.info>>.
- [ECDSA] Rijswijk-Deij, R., Sperotto, A., and A. Pras, "Making the Case for Elliptic Curves in DNSSEC", September 2015,
<<https://dl.acm.org/doi/10.1145/2831347.2831350>>.
- [FLAGDAY2020]
Various DNS software and service providers, "DNS Flag Day 2020", October 2020, <<https://dnsflagday.net/2020/>>.
- [FRAG_POISON]
Herzberg, A. and H. Shulman, "Fragmentation Considered Poisonous", May 2012,
<<https://u.cs.biu.ac.il/~herzbea/security/13-03-frag.pdf>>.
- [HUSTON] Huston, G., "Dealing with IPv6 fragmentation in the DNS", 22 August 2017, <<https://blog.apnic.net/2017/08/22/dealing-ipv6-fragmentation-dns/>>.
- [KSK_ROLLOVER_ARCHIVES]
Internet Corporation for Assigned Names and Numbers, "KSK Rollover List Archives", January 2019,
<<https://mm.icann.org/pipermail/ksk-rollover/2019-January/date.html>>.

- [LEWIS] Lewis, E., "2017 DNSSEC KSK Rollover", RIPE 74 Budapest, Hungary, 8 May 2017, <<https://ripe74.ripe.net/presentations/25-RIPE74-lewis-submission.pdf>>.
- [libpcap] Tcpdump/Libpcap, "Tcpdump and Libpcap", 7 May 2018, <<https://www.tcpdump.org>>.
- [NETALYZR] Kreibich, C., Weaver, N., Nechaev, B., and V. Paxson, "Netalyzer: Illuminating The Edge Network", 2010.
- [phrack] horizon, "Defeating Sniffers and Intrusion Detection Systems", December 1998, <<http://phrack.org/issues/54/10.html>>.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC0883] Mockapetris, P., "Domain names: Implementation specification", RFC 883, DOI 10.17487/RFC0883, November 1983, <<https://www.rfc-editor.org/info/rfc883>>.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <<https://www.rfc-editor.org/info/rfc1123>>.
- [RFC1536] Kumar, A., Postel, J., Neuman, C., Danzig, P., and S. Miller, "Common DNS Implementation Errors and Suggested Fixes", RFC 1536, DOI 10.17487/RFC1536, October 1993, <<https://www.rfc-editor.org/info/rfc1536>>.
- [RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, DOI 10.17487/RFC1995, August 1996, <<https://www.rfc-editor.org/info/rfc1995>>.
- [RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <<https://www.rfc-editor.org/info/rfc1996>>.

- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<https://www.rfc-editor.org/info/rfc2136>>.
- [RFC2541] Eastlake 3rd, D., "DNS Security Operational Considerations", RFC 2541, DOI 10.17487/RFC2541, March 1999, <<https://www.rfc-editor.org/info/rfc2541>>.
- [RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, DOI 10.17487/RFC2671, August 1999, <<https://www.rfc-editor.org/info/rfc2671>>.
- [RFC2694] Srisuresh, P., Tsirtsis, G., Akkiraju, P., and A. Heffernan, "DNS extensions to Network Address Translators (DNS_ALG)", RFC 2694, DOI 10.17487/RFC2694, September 1999, <<https://www.rfc-editor.org/info/rfc2694>>.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <<https://www.rfc-editor.org/info/rfc3022>>.
- [RFC3225] Conrad, D., "Indicating Resolver Support of DNSSEC", RFC 3225, DOI 10.17487/RFC3225, December 2001, <<https://www.rfc-editor.org/info/rfc3225>>.
- [RFC3226] Gudmundsson, O., "DNSSEC and IPv6 A6 aware server/resolver message size requirements", RFC 3226, DOI 10.17487/RFC3226, December 2001, <<https://www.rfc-editor.org/info/rfc3226>>.
- [RFC4472] Durand, A., Ihren, J., and P. Savola, "Operational Considerations and Issues with IPv6 DNS", RFC 4472, DOI 10.17487/RFC4472, April 2006, <<https://www.rfc-editor.org/info/rfc4472>>.
- [RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks", RFC 4953, DOI 10.17487/RFC4953, July 2007, <<https://www.rfc-editor.org/info/rfc4953>>.
- [RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <<https://www.rfc-editor.org/info/rfc4987>>.

- [RFC5358] Damas, J. and F. Neves, "Preventing Use of Recursive Nameservers in Reflector Attacks", BCP 140, RFC 5358, DOI 10.17487/RFC5358, October 2008, <<https://www.rfc-editor.org/info/rfc5358>>.
- [RFC5452] Hubert, A. and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers", RFC 5452, DOI 10.17487/RFC5452, January 2009, <<https://www.rfc-editor.org/info/rfc5452>>.
- [RFC5507] IAB, Faltstrom, P., Ed., Austein, R., Ed., and P. Koch, Ed., "Design Choices When Expanding the DNS", RFC 5507, DOI 10.17487/RFC5507, April 2009, <<https://www.rfc-editor.org/info/rfc5507>>.
- [RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <<https://www.rfc-editor.org/info/rfc5625>>.
- [RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, DOI 10.17487/RFC5927, July 2010, <<https://www.rfc-editor.org/info/rfc5927>>.
- [RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <<https://www.rfc-editor.org/info/rfc5936>>.
- [RFC5961] Ramaiah, A., Stewart, R., and M. Dalal, "Improving TCP's Robustness to Blind In-Window Attacks", RFC 5961, DOI 10.17487/RFC5961, August 2010, <<https://www.rfc-editor.org/info/rfc5961>>.
- [RFC5966] Bellis, R., "DNS Transport over TCP - Implementation Requirements", RFC 5966, DOI 10.17487/RFC5966, August 2010, <<https://www.rfc-editor.org/info/rfc5966>>.
- [RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <<https://www.rfc-editor.org/info/rfc6298>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.

- [RFC6950] Peterson, J., Kolkman, O., Tschofenig, H., and B. Aboba, "Architectural Considerations on Application Features in the DNS", RFC 6950, DOI 10.17487/RFC6950, October 2013, <<https://www.rfc-editor.org/info/rfc6950>>.
- [RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <<https://www.rfc-editor.org/info/rfc7413>>.
- [RFC7477] Hardaker, W., "Child-to-Parent Synchronization in DNS", RFC 7477, DOI 10.17487/RFC7477, March 2015, <<https://www.rfc-editor.org/info/rfc7477>>.
- [RFC7534] Abley, J. and W. Sotomayor, "AS112 Nameserver Operations", RFC 7534, DOI 10.17487/RFC7534, May 2015, <<https://www.rfc-editor.org/info/rfc7534>>.
- [RFC7720] Blanchet, M. and L-J. Liman, "DNS Root Name Service Protocol and Deployment Requirements", BCP 40, RFC 7720, DOI 10.17487/RFC7720, December 2015, <<https://www.rfc-editor.org/info/rfc7720>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC7901] Wouters, P., "CHAIN Query Requests in DNS", RFC 7901, DOI 10.17487/RFC7901, June 2016, <<https://www.rfc-editor.org/info/rfc7901>>.
- [RFC7918] Langley, A., Modadugu, N., and B. Moeller, "Transport Layer Security (TLS) False Start", RFC 7918, DOI 10.17487/RFC7918, August 2016, <<https://www.rfc-editor.org/info/rfc7918>>.
- [RFC8027] Hardaker, W., Gudmundsson, O., and S. Krishnaswamy, "DNSSEC Roadblock Avoidance", BCP 207, RFC 8027, DOI 10.17487/RFC8027, November 2016, <<https://www.rfc-editor.org/info/rfc8027>>.
- [RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <<https://www.rfc-editor.org/info/rfc8094>>.

- [RFC8162] Hoffman, P. and J. Schlyter, "Using Secure DNS to Associate Certificates with Domain Names for S/MIME", RFC 8162, DOI 10.17487/RFC8162, May 2017, <<https://www.rfc-editor.org/info/rfc8162>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8324] Klensin, J., "DNS Privacy, Authorization, Special Uses, Encoding, Characters, Matching, and Root Structure: Time for Another Look?", RFC 8324, DOI 10.17487/RFC8324, February 2018, <<https://www.rfc-editor.org/info/rfc8324>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8467] Mayrhofer, A., "Padding Policies for Extension Mechanisms for DNS (EDNS(0))", RFC 8467, DOI 10.17487/RFC8467, October 2018, <<https://www.rfc-editor.org/info/rfc8467>>.
- [RFC8482] Abley, J., Gudmundsson, O., Majkowski, M., and E. Hunt, "Providing Minimal-Sized Responses to DNS Queries That Have QTYPE=ANY", RFC 8482, DOI 10.17487/RFC8482, January 2019, <<https://www.rfc-editor.org/info/rfc8482>>.
- [RFC8483] Song, L., Ed., Liu, D., Vixie, P., Kato, A., and S. Kerr, "Yeti DNS Testbed", RFC 8483, DOI 10.17487/RFC8483, October 2018, <<https://www.rfc-editor.org/info/rfc8483>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.
- [RFC8490] Bellis, R., Cheshire, S., Dickinson, J., Dickinson, S., Lemon, T., and T. Pusateri, "DNS Stateful Operations", RFC 8490, DOI 10.17487/RFC8490, March 2019, <<https://www.rfc-editor.org/info/rfc8490>>.
- [RFC8501] Howard, L., "Reverse DNS in IPv6 for Internet Service Providers", RFC 8501, DOI 10.17487/RFC8501, November 2018, <<https://www.rfc-editor.org/info/rfc8501>>.
- [RFC8806] Kumari, W. and P. Hoffman, "Running a Root Server Local to a Resolver", RFC 8806, DOI 10.17487/RFC8806, June 2020, <<https://www.rfc-editor.org/info/rfc8806>>.

- [RFC8906] Andrews, M. and R. Bellis, "A Common Operational Problem in DNS Servers: Failure to Communicate", BCP 231, RFC 8906, DOI 10.17487/RFC8906, September 2020, <<https://www.rfc-editor.org/info/rfc8906>>.
- [RFC8932] Dickinson, S., Overeinder, B., van Rijswijk-Deij, R., and A. Mankin, "Recommendations for DNS Privacy Service Operators", BCP 232, RFC 8932, DOI 10.17487/RFC8932, October 2020, <<https://www.rfc-editor.org/info/rfc8932>>.
- [RFC8945] Dupont, F., Morris, S., Vixie, P., Eastlake 3rd, D., Gudmundsson, O., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", STD 93, RFC 8945, DOI 10.17487/RFC8945, November 2020, <<https://www.rfc-editor.org/info/rfc8945>>.
- [ROLL_YOUR_ROOT] Müller, M., Thomas, M., Wessels, D., Hardaker, W., Chung, T., Toorop, W., and R.v. Rijswijk-Deij, "Roll, Roll, Roll Your Root: A Comprehensive Analysis of the First Ever DNSSEC Root KSK Rollover", October 2019, <<https://dl.acm.org/doi/10.1145/3355369.3355570>>.
- [RRL] Vixie, P. and V. Schryver, "DNS Response Rate Limiting (DNS RRL)", ISC-TN 2012-1 Draft1, April 2012.
- [TDNS] Zhu, L., Heidemann, J., Wessels, D., Mankin, A., and N. Somaiya, "Connection-oriented DNS to Improve Privacy and Security", 2015.
- [TOYAMA] Toyama, K., Ishibashi, K., Ishino, M., Yoshimura, C., and K. Fujiwara, "DNS Anomalies and Their Impacts on DNS Cache Servers", NANOG 32 Reston, VA USA, 2004.
- [VERISIGN] Thomas, M. and D. Wessels, "An Analysis of TCP Traffic in Root Server DITL Data", DNS-OARC 2014 Fall Workshop Los Angeles, 2014.
- [WIKIPEDIA_TFO] Wikipedia, "TCP Fast Open", 4 May 2018, <https://en.wikipedia.org/wiki/TCP_Fast_Open>.

Appendix A. Standards Related to DNS Transport over TCP

This section enumerates all known IETF RFC documents that are currently of status Internet Standard, Draft Standard, Proposed Standard, Informational, Best Current Practice, or Experimental and either implicitly or explicitly make assumptions or statements about the use of TCP as a transport for the DNS germane to this document.

A.1. IETF RFC 1035 - DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION

The Internet Standard [RFC1035] is the base DNS specification that explicitly defines support for DNS over TCP.

A.2. IETF RFC 1536 - Common DNS Implementation Errors and Suggested Fixes

This Informational document [RFC1536] states UDP is the "chosen protocol for communication though TCP is used for zone transfers." That statement should now be considered in its historical context and is no longer a proper reflection of modern expectations.

A.3. IETF RFC 1995 - Incremental Zone Transfer in DNS

This Proposed Standard [RFC1995] documents the use of TCP as the fallback transport when IXFR responses do not fit into a single UDP response. As with AXFR, IXFR messages are typically delivered over TCP by default in practice.

A.4. IETF RFC 1996 - A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)

This Proposed Standard [RFC1996] suggests a primary server may decide to issue NOTIFY messages over TCP. In practice, NOTIFY messages are generally sent over UDP, but this specification leaves open the possibility that the choice of transport protocol is up to the primary server, and therefore a secondary server ought to be able to operate over both UDP and TCP.

A.5. IETF RFC 2181 - Clarifications to the DNS Specification

This Proposed Standard [RFC2181] includes clarifying text on how a client should react to the TC bit set on responses. It is advised that the response should be discarded and the query resent using TCP.

A.6. IETF RFC 2694 - DNS extensions to Network Address Translators (DNS_ALG)

This Informational document [RFC2694] enumerates considerations for network address translation (NAT) devices to properly handle DNS traffic. This document is noteworthy in its suggestion that "[t]ypically, TCP is used for AXFR requests," as further evidence that helps explain why DNS over TCP may have often been treated very differently than DNS over UDP in operational networks.

A.7. IETF RFC 3225 - Indicating Resolver Support of DNSSEC

This Proposed Standard [RFC3225] makes statements indicating DNS over TCP is "detrimental" as a result of increased traffic, latency, and server load. This document is a companion to the next document in the RFC series expressing the requirement for EDNS(0) support for DNSSEC.

A.8. IETF RFC 3226 - DNSSEC and IPv6 A6 aware server/resolver message size requirements

Although updated by later DNSSEC RFCs, the Proposed Standard [RFC3226] strongly argues in favor of UDP messages instead of TCP, largely for performance reasons. The document declares EDNS(0) a requirement for DNSSEC servers and advocates that packet fragmentation may be preferable to TCP in certain situations.

A.9. IETF RFC 4472 - Operational Considerations and Issues with IPv6 DNS

This Informational document [RFC4472] notes that IPv6 data may increase DNS responses beyond what would fit in a UDP message. Particularly noteworthy, perhaps less common today than when this document was written, it refers to implementations that truncate data without setting the TC bit to encourage the client to resend the query using TCP.

A.10. IETF RFC 5452 - Measures for Making DNS More Resilient against Forged Answers

This Informational document [RFC5452] arose as public DNS systems began to experience widespread abuse from spoofed queries, resulting in amplification and reflection attacks against unwitting victims. One of the leading justifications for supporting DNS over TCP to thwart these attacks is briefly described in this document's 9.3 Spoof Detection and Countermeasure section.

A.11. IETF RFC 5507 - Design Choices When Expanding the DNS

This Informational document [RFC5507] was largely an attempt to dissuade new DNS data types from overloading the TXT resource record type. In so doing it summarizes the conventional wisdom of DNS design and implementation practices. The authors suggest TCP overhead and stateful properties pose challenges compared to UDP, and imply that UDP is generally preferred for performance and robustness.

A.12. IETF RFC 5625 - DNS Proxy Implementation Guidelines

This Best Current Practice document [RFC5625] provides DNS proxy implementation guidance including the mandate that a proxy "MUST [...] be prepared to receive and forward queries over TCP" even though it suggests historically TCP transport has not been strictly mandatory in stub resolvers or recursive servers.

A.13. IETF RFC 5936 - DNS Zone Transfer Protocol (AXFR)

This Proposed Standard [RFC5936] provides a detailed specification for the zone transfer protocol, as originally outlined in the early DNS standards. AXFR operation is limited to TCP and not specified for UDP. This document discusses TCP usage at length.

A.14. IETF RFC 7534 - AS112 Nameserver Operations

[RFC7534] is an Informational document enumerating the requirements for operation of AS112 project DNS servers. New AS112 nodes are tested for their ability to provide service on both UDP and TCP transports, with the implication that TCP service is an expected part of normal operations.

A.15. IETF RFC 6762 - Multicast DNS

In this Proposed Standard [RFC6762], the TC bit is deemed to have essentially the same meaning as described in the original DNS specifications. That is, if a response with the TC bit set is received, "[...] the querier SHOULD reissue its query using TCP in order to receive the larger response."

A.16. IETF RFC 6891 - Extension Mechanisms for DNS (EDNS(0))

This Internet Standard [RFC6891] helped slow the use of and need for DNS over TCP messages. This document highlights concerns over server load and scalability in widespread use of DNS over TCP.

A.17. IETF RFC 6950 - Architectural Considerations on Application Features in the DNS

An Informational document [RFC6950] that draws attention to large data in the DNS. TCP is referenced in the context as a common fallback mechanism and counter to some spoofing attacks.

A.18. IETF RFC 7477 - Child-to-Parent Synchronization in DNS

This Proposed Standard [RFC7477] specifies a RRTYPE and protocol to signal and synchronize NS, A, and AAAA resource record changes from a child to parent zone. Since this protocol may require multiple requests and responses, it recommends utilizing DNS over TCP to ensure the conversation takes place between a consistent pair of end nodes.

A.19. IETF RFC 7720 - DNS Root Name Service Protocol and Deployment Requirements

This Best Current Practice [RFC7720] declares root name service "MUST support UDP [RFC0768] and TCP [RFC0793] transport of DNS queries and responses."

A.20. IETF RFC 7766 - DNS Transport over TCP - Implementation Requirements

This Proposed Standard [RFC7766] instructs DNS implementers to provide support for carrying DNS over TCP messages in their software, and might be considered the direct ancestor of this operational requirements document. The implementation requirements document codifies mandatory support for DNS over TCP in compliant DNS software, but makes no recommendations to operators, which we seek to address here.

A.21. IETF RFC 7828 - The edns-tcp-keepalive EDNS(0) Option

This Proposed Standard [RFC7828] defines an EDNS(0) option to negotiate an idle timeout value for long-lived DNS over TCP connections. Consequently, this document is only applicable and relevant to DNS over TCP sessions and between implementations that support this option.

A.22. IETF RFC 7858 - Specification for DNS over Transport Layer Security (TLS)

This Proposed Standard [RFC7858] defines a method for putting DNS messages into a TCP-based encrypted channel using TLS. This specification is noteworthy for explicitly targeting the stub-to-recursive traffic, but does not preclude its application from recursive-to-authoritative traffic.

A.23. IETF RFC 7873 - Domain Name System (DNS) Cookies

This Proposed Standard [RFC7873] describes an EDNS(0) option to provide additional protection against query and answer forgery. This specification mentions DNS over TCP as an alternative mechanism when DNS Cookies are not available. The specification does make mention of DNS over TCP processing in two specific situations. In one, when a server receives only a client cookie in a request, the server should consider whether the request arrived over TCP and if so, it should consider accepting TCP as sufficient to authenticate the request and respond accordingly. In another, when a client receives a BADCOOKIE reply using a fresh server cookie, the client should retry using TCP as the transport.

A.24. IETF RFC 7901 - CHAIN Query Requests in DNS

This Experimental specification [RFC7901] describes an EDNS(0) option that can be used by a security-aware validating resolver to request and obtain a complete DNSSEC validation path for any single query. This document requires the use of DNS over TCP or a source IP address verified transport mechanism such as EDNS-COOKIE [RFC7873].

A.25. IETF RFC 8027 - DNSSEC Roadblock Avoidance

This Best Current Practice [RFC8027] details observed problems with DNSSEC deployment and mitigation techniques. Network traffic blocking and restrictions, including DNS over TCP messages, are highlighted as one reason for DNSSEC deployment issues. While this document suggests these sorts of problems are due to "non-compliant infrastructure", the scope of the document is limited to detection and mitigation techniques to avoid so-called DNSSEC roadblocks.

A.26. IETF RFC 8094 - DNS over Datagram Transport Layer Security (DTLS)

This Experimental specification [RFC8094] details a protocol that uses a datagram transport (UDP), but stipulates that "DNS clients and servers that implement DNS over DTLS MUST also implement DNS over TLS in order to provide privacy for clients that desire Strict Privacy [...]." This requirement implies DNS over TCP must be supported in case the message size is larger than the path MTU.

A.27. IETF RFC 8162 - Using Secure DNS to Associate Certificates with Domain Names for S/MIME

This Experimental specification [RFC8162] describes a technique to authenticate user X.509 certificates in an S/MIME system via the DNS. The document points out that the new experimental resource record types are expected to carry large payloads, resulting in the suggestion that "applications SHOULD use TCP -- not UDP -- to perform queries for the SMIMEA resource record."

A.28. IETF RFC 8324 - DNS Privacy, Authorization, Special Uses, Encoding, Characters, Matching, and Root Structure: Time for Another Look?

An Informational document [RFC8324] that briefly discusses the common role and challenges of DNS over TCP throughout the history of DNS.

A.29. IETF RFC 8467 - Padding Policies for Extension Mechanisms for DNS (EDNS(0))

An Experimental document [RFC8467] reminds implementers to consider the underlying transport protocol (e.g. TCP) when calculating the padding length when artificially increasing the DNS message size with an EDNS(0) padding option.

A.30. IETF RFC 8482 - Providing Minimal-Sized Responses to DNS Queries That Have QTYPE=ANY

[RFC8482] is a Proposed Standard that describes alternative ways that DNS servers can respond to queries of type ANY, which are sometimes used to provide amplification in DDoS attacks. The specification notes that responders may behave differently, depending on the transport. For example, minimal-sized responses may be used over UDP transport, while full responses may be given over TCP.

A.31. IETF RFC 8483 - Yeti DNS Testbed

This Informational document [RFC8483] describes a testbed environment that highlights some DNS over TCP behaviors, including issues involving packet fragmentation and operational requirements for TCP stream assembly in order to conduct DNS measurement and analysis.

A.32. IETF RFC 8484 - DNS Queries over HTTPS (DoH)

This Proposed Standard [RFC8484] defines a protocol for sending DNS queries and responses over HTTPS. This specification assumes TLS and TCP for the underlying security and transport layers, respectively. Self-described as a technique that more closely resembles a tunneling mechanism, DoH nevertheless likely implies DNS over TCP in some sense, if not directly.

A.33. IETF RFC 8490 - DNS Stateful Operations

This Proposed Standard [RFC8490] updates the base protocol specification with a new OPCODE to help manage stateful operations in persistent sessions, such as those that might be used by DNS over TCP.

A.34. IETF RFC 8501 - Reverse DNS in IPv6 for Internet Service Providers

This Informational document [RFC8501] identifies potential operational challenges with Dynamic DNS including denial-of-service threats. The document suggests TCP may provide some advantages, but that updating hosts would need to be explicitly configured to use TCP instead of UDP.

A.35. IETF RFC 8806 - Running a Root Server Local to a Resolver

This Informational document [RFC8806] describes how to obtain and operate a local copy of the root zone with examples showing how to pull from authoritative sources using a DNS over TCP zone transfer.

A.36. IETF RFC 8906 - A Common Operational Problem in DNS Servers: Failure to Communicate

This Best Current Practice document [RFC8906] discusses a number of DNS operational failure scenarios and how to avoid them. This includes discussions involving DNS over TCP queries, EDNS over TCP, and a testing methodology that includes a section on verifying DNS over TCP functionality.

A.37. IETF RFC 8932 - Recommendations for DNS Privacy Service Operators

This Best Current Practice document [RFC8932] presents privacy considerations to DNS privacy service operators. These mechanisms sometimes include the use of TCP and are therefore susceptible to information leakage such as TCP-based fingerprinting. This document also references a draft version of this document.

A.38. IETF RFC 8945 - Secret Key Transaction Authentication for DNS (TSIG)

This Internet Standard [RFC8945] recommends a client use TCP if truncated TSIG messages are received.

Authors' Addresses

John Kristoff
DataPlane.org
Chicago, IL 60605
United States of America

Phone: +1 312 493 0305
Email: jtk@dataplane.org
URI: <https://dataplane.org/jtk/>

Duane Wessels
Verisign
12061 Bluemont Way
Reston, VA 20190
United States of America

Phone: +1 703 948 3200
Email: dwessels@verisign.com
URI: <https://verisign.com>

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: April 18, 2021

D. Wessels
P. Barber
Verisign
M. Weinberg
Amazon
W. Kumari
Google
W. Hardaker
USC/ISI
October 15, 2020

Message Digest for DNS Zones
draft-ietf-dnsop-dns-zone-digest-14

Abstract

This document describes a protocol and new DNS Resource Record that provides a cryptographic message digest over DNS zone data at rest. The ZONEMD Resource Record conveys the digest data in the zone itself. When used in combination with DNSSEC, ZONEMD allows recipients to verify the zone contents for data integrity and origin authenticity. This provides assurance that received zone data matches published data, regardless of how the zone data has been transmitted and received. When used without DNSSEC, ZONEMD functions as a checksum, guarding only against unintentional changes.

ZONEMD does not replace DNSSEC. Whereas DNSSEC protects individual RRSets (DNS data with fine granularity), ZONEMD protects a zone's data as a whole, whether consumed by authoritative name servers, recursive name servers, or any other applications.

As specified herein, ZONEMD is impractical for large, dynamic zones due to the time and resources required for digest calculation. However, The ZONEMD record is extensible so that new digest schemes may be added in the future to support large, dynamic zones.

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 April 18, 2021.

Copyright Notice

Copyright (c) 2020 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 (<https://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. Introduction	3
1.1. Motivation	4
1.2. Alternative Approaches	4
1.3. Design Overview	6
1.4. Use Cases	6
1.4.1. Root Zone	6
1.4.2. Providers, Secondaries, and Anycast	7
1.4.3. Response Policy Zones	7
1.4.4. Centralized Zone Data Service	7
1.4.5. General Purpose Comparison Check	7
1.5. Terminology	8
2. The ZONEMD Resource Record	8
2.1. Non-apex ZONEMD Records	8
2.2. ZONEMD RDATA Wire Format	8
2.2.1. The Serial Field	9
2.2.2. The Scheme Field	9
2.2.3. The Hash Algorithm Field	9
2.2.4. The Digest Field	10
2.3. ZONEMD Presentation Format	10
2.4. ZONEMD Example	10
2.5. Including ZONEMD RRs in a Zone	10
3. Calculating the Digest	11
3.1. Add ZONEMD Placeholder	11
3.2. Optionally Sign the Zone	12

3.3.	Scheme-Specific Processing	12
3.3.1.	The SIMPLE Scheme	12
3.3.1.1.	SIMPLE Scheme Inclusion/Exclusion Rules	12
3.3.1.2.	SIMPLE Scheme Digest Calculation	13
3.4.	Update ZONEMD RR	13
4.	Verifying Zone Digest	13
5.	IANA Considerations	15
5.1.	ZONEMD Rrtype	15
5.2.	ZONEMD Scheme	15
5.3.	ZONEMD Hash Algorithm	16
6.	Security Considerations	16
6.1.	Using Zone Digest Without DNSSEC	16
6.2.	Attacks Against the Zone Digest	16
6.3.	Use of Multiple ZONEMD Hash Algorithms	17
6.4.	DNSSEC Timing Considerations	17
6.5.	Attacks Utilizing ZONEMD Queries	17
6.6.	Resilience and Fragility	18
7.	Performance Considerations	18
7.1.	SIMPLE SHA384	18
8.	Privacy Considerations	19
9.	Acknowledgments	19
10.	Change Log	19
11.	References	27
11.1.	Normative References	27
11.2.	Informative References	27
Appendix A.	Example Zones With Digests	30
A.1.	Simple EXAMPLE Zone	30
A.2.	Complex EXAMPLE Zone	30
A.3.	EXAMPLE Zone with multiple digests	31
A.4.	The URI.ARPA Zone	32
A.5.	The ROOT-SERVERS.NET Zone	35
Appendix B.	Implementation Status	37
B.1.	Authors' Implementation	37
B.2.	Shane Kerr's Implementation	37
B.3.	NIC Chile Labs Implementation	38
Authors'	Addresses	38

1. Introduction

In the DNS, a zone is the collection of authoritative resource records (RRs) sharing a common origin ([RFC8499]). Zones are often stored as files in the so-called master file format [RFC1034]. Zones are generally distributed among name servers using the AXFR (zone transfer [RFC5936]), and IXFR (incremental zone transfer [RFC1995]) protocols. They can also be distributed outside of the DNS, with any file transfer protocol such as FTP, HTTP, and rsync, or even as email attachments. Currently, there is no standard way to compute a hash or message digest for a stand-alone zone.

This document specifies an RR type that provides a cryptographic message digest of the data in a zone. It allows a receiver of the zone to verify the zone's integrity and authenticity when used in combination with DNSSEC. The digest RR is a part of the zone itself, allowing verification of the zone, no matter how it is transmitted. The digest uses the wire format of zone data in a canonical ordering. Thus, it is independent of presentation format, such as whitespace, capitalization, and comments.

This specification is OPTIONAL to implement by both publishers and consumers of zone data.

1.1. Motivation

The primary motivation for this protocol enhancement is the desire to verify the data integrity and origin authenticity of a stand-alone zone, regardless of how it is transmitted. A consumer of zone data should be able to verify that it is as-published by the zone operator.

Note, however, that integrity and authenticity can only be assured when the zone is signed. DNSSEC provides three strong security guarantees relevant to this protocol:

1. whether or not to expect DNSSEC records in the zone,
2. whether or not to expect a ZONEMD record in a signed zone, and
3. whether or not the ZONEMD record has been altered since it was signed.

A secondary motivation is to provide the equivalent of a checksum, allowing a zone recipient to check for unintended changes and operational errors, such as accidental truncation.

1.2. Alternative Approaches

One approach to preventing data tampering and corruption is to secure the distribution channel. The DNS has a number of features that are already used for channel security. Perhaps the most widely used is DNS transaction signatures (TSIG [RFC2845]). TSIG uses shared secret keys and a message digest to protect individual query and response messages. It is generally used to authenticate and validate UPDATE [RFC2136], AXFR [RFC5936], and IXFR [RFC1995] messages.

DNS Request and Transaction Signatures (SIG(0) [RFC2931]) is another protocol extension that authenticates individual DNS transactions. Whereas SIG records normally cover specific RR types, SIG(0) is used

to sign an entire DNS message. Unlike TSIG, SIG(0) uses public key cryptography rather than shared secrets.

The Transport Layer Security protocol suite also provides channel security. The DPRIVE working group is in the process of specifying DNS Zone Transfer-over-TLS [I-D.ietf-dprive-xfr-over-tls]. One can also easily imagine the distribution of zones over HTTPS-enabled web servers, as well as DNS-over-HTTPS [RFC8484].

Unfortunately, the protections provided by these channel security techniques are (in practice) ephemeral and are not retained after the data transfer is complete. They ensure that the client receives the data from the expected server, and that the data sent by the server is not modified during transmission. However, they do not guarantee that the server transmits the data as originally published, and do not provide any methods to verify data that is read after transmission is complete. For example, a name server loading saved zone data upon restart cannot guarantee that the on-disk data has not been modified. Such modification could be the result of an accidental corruption of the file, or perhaps an incompletely saved file [disk-full-failure]. For these reasons, it is preferable to protect the integrity of the data itself.

Why not simply rely on DNSSEC, which provides certain data security guarantees? For zones that are signed, a recipient could validate all of the signed RRSets. Additionally, denial-of-existence records prove that RRSets have not been added or removed. However, delegations (non-apex NS records) are not signed by DNSSEC, and neither are any glue records. ZONEMD protects the integrity of delegation, glue, and other records that are not otherwise covered by DNSSEC. Furthermore, zones that employ NSEC3 with opt-out [RFC5155] are susceptible to the removal or addition of names between the signed nodes. Whereas DNSSEC primarily protects consumers of DNS response messages, this protocol protects consumers of zones.

There are existing tools and protocols that provide data security, such as OpenPGP [RFC4880] and S/MIME [RFC5751]. In fact, the internic.net site publishes PGP signatures alongside the root zone and other files available there. However, this is a detached signature with no strong association to the corresponding zone file other than its timestamp. Non-detached signatures are, of course, possible, but these necessarily change the format of the file being distributed; a zone signed with OpenPGP or S/MIME no longer looks like a DNS zone and could not directly be loaded into a name server. Once loaded the signature data is lost, so it cannot be further propagated.

It seems the desire for data security in DNS zones was envisioned as far back as 1997. [RFC2065] is an obsoleted specification of the first generation DNSSEC Security Extensions. It describes a zone transfer signature, identified as the AXFR SIG, which is similar to the technique proposed by this document. That is, it proposes ordering all (signed) RRSsets in a zone, hashing their contents, and then signing the zone hash. The AXFR SIG is described only for use during zone transfers. It did not postulate the need to validate zone data distributed outside of the DNS. Furthermore, its successor, [RFC2535], omits the AXFR SIG, while at the same time introducing an IXFR SIG.

1.3. Design Overview

This document specifies a new Resource Record type to convey a message digest of the content of a zone. The digest is calculated at the time of zone publication. If the zone is signed with DNSSEC, any modifications of the digest can be detected. The procedures for digest calculation and DNSSEC signing are similar. Both require data to be processed in a well-defined order and format. It may be possible to perform DNSSEC signing and digest calculation in parallel.

The zone digest is designed to be used on zones that have infrequent updates. As specified herein, the digest is re-calculated over the entire zone content each time the zone is updated. This specification does not provide an efficient mechanism for updating the digest on incremental updates of zone data. It is, however, extensible so that future schemes may be defined to support efficient incremental digest updates.

It is expected that verification of a zone digest will be implemented in name server software. That is, a name server can verify the zone data it was given and refuse to serve a zone which fails verification. For signed zones, the name server needs a trust anchor to perform DNSSEC validation. For signed non-root zones, the name server may need to send queries to validate a chain of trust. Digest verification could also be performed externally.

1.4. Use Cases

1.4.1. Root Zone

The root zone [InterNIC] is one of the most widely distributed DNS zone on the Internet, served by more than 1000 separate instances [RootServers] at the time of this writing. Additionally, many organizations configure their own name servers to serve the root zone locally. Reasons for doing so include privacy and reduced access

time. [RFC8806] describes one way to do this. As the root zone spreads beyond its traditional deployment boundaries, the verification of the completeness of the zone contents becomes more important.

1.4.2. Providers, Secondaries, and Anycast

Since its very early days, the developers of the DNS recognized the importance of secondary name servers and service diversity. However, modern DNS service has complex provisioning which includes multiple third-party providers ([RFC8901]) and hundreds of anycast instances ([RFC3258]). Instead of a simple primary-to-secondary zone distribution system, today it is possible to have multiple levels, multiple parties, and multiple protocols involved in the distribution of zone data. This complexity introduces new places for problems to arise. The zone digest protects the integrity of data that flows through such systems.

1.4.3. Response Policy Zones

A Response Policy Zone (RPZ) is "a mechanism to introduce a customized policy in Domain Name System servers, so that recursive resolvers return possibly modified results" [RPZ]. The policy information is carried inside specially constructed DNS zones. A number of companies provide RPZ feeds, which are consumed by name server and firewall products. While RPZ zones can be signed with DNSSEC, the data is not queried directly, and would not be subject to DNSSEC validation.

1.4.4. Centralized Zone Data Service

ICANN operates the Centralized Zone Data Service [CZDS], which is a repository of top-level domain zone files. Users that have been granted access are then able to download zone data. Adding a zone digest to these would provide CZDS users with assurances that the data has not been modified between origination and retrieval. Note that ZONEMD could be added to zone data supplied to CZDS without requiring it to be present in the zone data served by production name servers, since the digest is inherently attached to the specific copy of the zone.

1.4.5. General Purpose Comparison Check

Since the zone digest calculation does not depend on presentation format, it could be used to compare multiple copies of a zone received from different sources, or copies generated by different processes. In this case, it serves as a checksum and can be useful even for unsigned zones.

1.5. Terminology

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.

The terms Private Use, Reserved, Unassigned, and Specification Required are to be interpreted as defined in [RFC8126].

2. The ZONEMD Resource Record

This section describes the ZONEMD Resource Record, including its fields, wire format, and presentation format. The Type value for the ZONEMD RR is 63. The ZONEMD RR is class independent. The RDATA of the resource record consists of four fields: Serial, Scheme, Hash Algorithm, and Digest.

2.1. Non-apex ZONEMD Records

This document specifies ZONEMD RRs located at the zone apex. Non-apex ZONEMD RRs are not forbidden, but have no meaning in this specification. Non-apex ZONEMD RRs MUST NOT be used for verification.

During digest calculation, non-apex ZONEMD RRs are treated as ordinary RRs. They are digested as-is and the RR is not replaced by a placeholder RR.

Unless explicitly stated otherwise, "ZONEMD" always refers to apex records throughout this document.

2.2. ZONEMD RDATA Wire Format

The ZONEMD RDATA wire format is encoded as follows:

```

      1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Serial                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Scheme   | Hash Algorithm |                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Digest                               |
/                                                         /
/                                                         /
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.2.1. The Serial Field

The Serial field is a 32-bit unsigned integer in network byte order. It is the serial number from the zone's SOA record ([RFC1035] section 3.3.13) for which the zone digest was generated.

It is included here to clearly bind the ZONEMD RR to a particular version of the zone's content. Without the serial number, a stand-alone ZONEMD digest has no obvious association to any particular instance of a zone.

2.2.2. The Scheme Field

The Scheme field is an 8-bit unsigned integer that identifies the methods by which data is collated and presented as input to the hashing function.

Herein, SIMPLE, with Scheme value 1, is the only standardized Scheme defined for ZONEMD records and it MUST be supported by implementations. The Scheme registry is further described in Section 5.

Scheme values 240-254 are allocated for Private Use.

2.2.3. The Hash Algorithm Field

The Hash Algorithm field is an 8-bit unsigned integer that identifies the cryptographic hash algorithm used to construct the digest.

Herein, SHA384 [RFC6234], with Hash Algorithm value 1, is the only standardized Hash Algorithm defined for ZONEMD records that MUST be supported by implementations. When SHA384 is used, the size of the Digest field is 48 octets. The result of the SHA384 digest algorithm MUST NOT be truncated, and the entire 48 octet digest is published in the ZONEMD record.

SHA512 [RFC6234], with Hash Algorithm value 2, is also defined for ZONEMD records, and SHOULD be supported by implementations. When SHA512 is used, the size of the Digest field is 64 octets. The result of the SHA512 digest algorithm MUST NOT be truncated, and the entire 64 octet digest is published in the ZONEMD record.

Hash Algorithm values 240-254 are allocated for Private Use.

The Hash Algorithm registry is further described in Section 5.

2.2.4. The Digest Field

The Digest field is a variable-length sequence of octets containing the output of the hash algorithm. The length of the Digest field is determined by deducting the fixed size of the Serial, Scheme, and Hash Algorithm fields from the RDATA size in the ZONEMD RR header.

The Digest field MUST NOT be shorter than 12 octets. Digests for the SHA384 and SHA512 hash algorithms specified herein are never truncated. Digests for future hash algorithms MAY be truncated, but MUST NOT be truncated to a length that results in less than 96-bits (12 octets) of equivalent strength.

Section 3 describes how to calculate the digest for a zone. Section 4 describes how to use the digest to verify the contents of a zone.

2.3. ZONEMD Presentation Format

The presentation format of the RDATA portion is as follows:

The Serial field is represented as an unsigned decimal integer.

The Scheme field is represented as an unsigned decimal integer.

The Hash Algorithm field is represented as an unsigned decimal integer.

The Digest is represented as a sequence of case-insensitive hexadecimal digits. Whitespace is allowed within the hexadecimal text.

2.4. ZONEMD Example

The following example shows a ZONEMD RR in presentation format:

```
example.com. 86400 IN ZONEMD 2018031500 1 1 (
  FEBE3D4CE2EC2FFA4BA99D46CD69D6D29711E55217057BEE
  7EB1A7B641A47BA7FED2DD5B97AE499FAFA4F22C6BD647DE )
```

2.5. Including ZONEMD RRs in a Zone

The zone operator chooses an appropriate hash algorithm and scheme, and includes the calculated zone digest in the apex ZONEMD RRset. The zone operator MAY choose any of the defined hash algorithms and schemes, including the private use code points.

The ZONEMD RRS_{et} MAY contain multiple records to support algorithm agility [RFC7696]. [RFC Editor: change that to BCP 201] When multiple ZONEMD RRs are present, each MUST specify a unique Scheme and Hash Algorithm tuple. It is RECOMMENDED that a zone include only one ZONEMD RR, unless the zone operator is in the process of transitioning to a new scheme or hash algorithm.

3. Calculating the Digest

The algorithm described in this section is designed for the common case of offline DNSSEC signing. Slight deviations may be permitted or necessary in other situations, such as with unsigned zones or online DNSSEC signing. Implementations that deviate from the described algorithm are advised to ensure that it produces ZONEMD RRs, signatures, and denial-of-existence records that are identical to the ones generated by this procedure.

3.1. Add ZONEMD Placeholder

In preparation for calculating the zone digest(s), any existing ZONEMD records (and covering RRSIGs) at the zone apex are first deleted.

Prior to calculation of the digest, and prior to signing with DNSSEC, one or more placeholder ZONEMD records are added to the zone apex. This ensures that denial-of-existence (NSEC, NSEC3) records are created correctly if the zone is signed with DNSSEC. If placeholders were not added prior to signing, the later addition of ZONEMD records would also require updating the Type Bit Maps field of any apex NSEC/NSEC3 RRs, which then invalidates the calculated digest value.

When multiple ZONEMD RRs are published in the zone, e.g., during an algorithm rollover, each MUST specify a unique Scheme and Hash Algorithm tuple.

It is RECOMMENDED that the TTL of the ZONEMD record match the TTL of the SOA. However, the TTL of the ZONEMD record may be safely ignored during verification in all cases.

In the placeholder record, the Serial field is set to the current SOA Serial. The Scheme field is set to the value for the chosen collation scheme. The Hash Algorithm field is set to the value for the chosen hash algorithm. Since apex ZONEMD records are excluded from digest calculation, the value of the Digest field does not matter at this point in the process.

3.2. Optionally Sign the Zone

Following addition of placeholder records, the zone may be signed with DNSSEC. When the digest calculation is complete, and the ZONEMD record is updated, the signature(s) for the ZONEMD RRSets MUST be recalculated and updated as well. Therefore, the signer is not required to calculate a signature over the placeholder record at this step in the process, but it is harmless to do so.

3.3. Scheme-Specific Processing

Herein, only the SIMPLE collation scheme is defined. Additional schemes may be defined in future updates to this document.

3.3.1. The SIMPLE Scheme

For the SIMPLE scheme, the digest is calculated over the zone as a whole. This means that a change to a single RR in the zone requires iterating over all RRs in the zone to recalculate the digest. SIMPLE is a good choice for zones that are small and/or stable, but probably not good for zones that are large and/or dynamic.

Calculation of a zone digest requires RRs to be processed in a consistent format and ordering. This specification uses DNSSEC's canonical on-the-wire RR format (without name compression) and ordering as specified in Sections 6.1, 6.2, and 6.3 of [RFC4034] with the additional provision that RRSets having the same owner name MUST be numerically ordered, in ascending order, by their numeric RR TYPE.

3.3.1.1. SIMPLE Scheme Inclusion/Exclusion Rules

When iterating over records in the zone, the following inclusion/exclusion rules apply:

- o All records in the zone, including glue records, MUST be included, unless excluded by a subsequent rule.
- o Occluded data ([RFC5936] Section 3.5) MUST be included.
- o If there are duplicate RRs with equal owner, class, type, and RDATA, only one instance is included ([RFC4034] Section 6.3), and the duplicates MUST be omitted.
- o The placeholder apex ZONEMD RR(s) MUST NOT be included.
- o If the zone is signed, DNSSEC RRs MUST be included, except:

- o The RRSIG covering the apex ZONEMD RRSet MUST NOT be included because the RRSIG will be updated after all digests have been calculated.

3.3.1.2. SIMPLE Scheme Digest Calculation

A zone digest using the SIMPLE scheme is calculated by concatenating all RRs in the zone, in the format and order described in Section 3.3.1 subject to the inclusion/exclusion rules described in Section 3.3.1.1, and then applying the chosen hash algorithm:

```
digest = hash( RR(1) | RR(2) | RR(3) | ... )
```

where "|" denotes concatenation.

3.4. Update ZONEMD RR

The calculated zone digest is inserted into the placeholder ZONEMD RR. Repeat for each digest if multiple digests are to be published.

If the zone is signed with DNSSEC, the RRSIG record(s) covering the ZONEMD RRSet MUST then be added or updated. Because the ZONEMD placeholder was added prior to signing, the zone will already have the appropriate denial-of-existence (NSEC, NSEC3) records.

Some DNSSEC implementations (especially "online signing") might update the SOA serial number whenever a new signature is made. To preserve the calculated digest, generation of a ZONEMD signature MUST NOT also result in a change to the SOA serial number. The ZONEMD RR and the matching SOA MUST be published at the same time.

4. Verifying Zone Digest

The recipient of a zone that has a ZONEMD RR verifies the zone by calculating the digest as follows. If multiple ZONEMD RRs are present in the zone, e.g., during an algorithm rollover, a match using any one of the recipient's supported Schemes and Hash Algorithms is sufficient to verify the zone. The verifier MAY ignore a ZONEMD RR if its Scheme and Hash Algorithm violates local policy.

1. The verifier MUST first determine whether or not to expect DNSSEC records in the zone. By examining locally configured trust anchors, and, if necessary, querying for and validating DS RRs in the parent zone, the verifier knows whether or not the zone to be verified should include DNSSEC keys and signatures. For zones where signatures are not expected, or if DNSSEC validation is not performed, digest verification continues at step 4 below.

2. For zones where signatures are expected, the existence of the apex ZONEMD record MUST be validated. If the DNSSEC data proves the ZONEMD RRSet does not exist, digest verification cannot occur. If the DNSSEC data proves the ZONEMD does exist, but is not found in the zone, digest verification MUST NOT be considered successful.
3. For zones where signatures are expected, the SOA and ZONEMD RRsets MUST have valid signatures, chaining up to a trust anchor. If DNSSEC validation of the SOA or ZONEMD RRsets fails, digest verification MUST NOT be considered successful.
4. When multiple ZONEMD RRs are present, each MUST specify a unique Scheme and Hash Algorithm tuple. If the ZONEMD RRSet contains more than one RR with the same Scheme and Hash Algorithm, digest verification for those ZONEMD RRs MUST NOT be considered successful.
5. Loop over all apex ZONEMD RRs and perform the following steps:
 - A. The SOA Serial field MUST exactly match the ZONEMD Serial field. If the fields do not match, digest verification MUST NOT be considered successful with this ZONEMD RR.
 - B. The Scheme field MUST be checked. If the verifier does not support the given scheme, verification MUST NOT be considered successful with this ZONEMD RR.
 - C. The Hash Algorithm field MUST be checked. If the verifier does not support the given hash algorithm, verification MUST NOT be considered successful with this ZONEMD RR.
 - D. The Digest field size MUST be checked. If the size of the given Digest field is smaller than 12 octets, or if the size is not equal to the size expected for the corresponding Hash Algorithm, verification MUST NOT be considered successful with this ZONEMD RR.
 - E. The zone digest is computed over the zone data as described in Section 3.3, using the Scheme and Hash Algorithm for the current ZONEMD RR.
 - F. The computed digest is compared to the received digest. If the two digest values match, verification is considered successful. Otherwise, verification MUST NOT be considered successful for this ZONEMD RR.

Each time zone verification is performed, the verifier SHOULD report the status as either successful or unsuccessful. When unsuccessful, the verifier SHOULD report the reason(s) that verification did not succeed.

5. IANA Considerations

5.1. ZONEMD RRtype

This document defines a new DNS RR type, ZONEMD, whose value 63 has been allocated by IANA from the "Resource Record (RR) TYPEs" subregistry of the "Domain Name System (DNS) Parameters" registry:

Type: ZONEMD

Value: 63

Meaning: Message Digest Over Zone Data

Reference: [this document]

5.2. ZONEMD Scheme

IANA is requested to create a new registry on the "Domain Name System (DNS) Parameters" web page as follows:

Registry Name: ZONEMD Schemes

Registration Procedure: Specification Required

Reference: [this document]

Value	Description	Mnemonic	Reference
0	Reserved		
1	Simple ZONEMD collation	SIMPLE	[this document]
2-239	Unassigned		
240-254	Private Use	N/A	[this document]
255	Reserved		

Table 1: ZONEMD Scheme Registry

5.3. ZONEMD Hash Algorithm

IANA is requested to create a new registry on the "Domain Name System (DNS) Parameters" web page as follows:

Registry Name: ZONEMD Hash Algorithms

Registration Procedure: Specification Required

Reference: [this document]

Value	Description	Mnemonic	Reference
0	Reserved		
1	SHA-384	SHA384	[this document]
2	SHA-512	SHA512	[this document]
3-239	Unassigned		
240-254	Private Use	N/A	[his document]
255	Reserved		

Table 2: ZONEMD Hash Algorithm Registry

6. Security Considerations

6.1. Using Zone Digest Without DNSSEC

Users of ZONEMD with unsigned zones are advised that it provides no real protection against attacks. While zone digests can be used in the absence of DNSSEC, this only provides protection against accidental zone corruption, such as transmission errors and truncation. When used in this manner, it effectively serves only as a checksum. For zones not signed with DNSSEC, an attacker can make any zone modifications appear to be valid by recomputing Digest field of a ZONEMD RR.

6.2. Attacks Against the Zone Digest

An attacker, whose goal is to modify zone content before it is used by the victim, may consider a number of different approaches.

The attacker might perform a downgrade attack to an unsigned zone. This is why Section 4 talks about determining whether or not to expect DNSSEC signatures for the zone in step 1.

The attacker might perform a downgrade attack by removing one or more ZONEMD records. Such a removal is detectable only with DNSSEC

validation and is why Section 4 talks about checking denial-of-existence proofs in step 2 and signature validation in step 3.

The attacker might alter the Scheme, Hash Algorithm, or Digest fields of the ZONEMD record. Such modifications are detectable only with DNSSEC validation.

As stated in [RFC7696], cryptographic algorithms age and become weaker as cryptanalysis techniques and computing resources improve with time. Implementors and publishers of zone digests should anticipate the need for algorithm agility on long timescales.

6.3. Use of Multiple ZONEMD Hash Algorithms

When a zone publishes multiple ZONEMD RRs, the overall security is only as good as the weakest hash algorithm in use. For this reason, Section 2 recommends only publishing multiple ZONEMD RRs when transitioning to a new scheme or hash algorithm. Once the transition is complete, the old scheme or hash algorithm should be removed from the ZONEMD RRSets.

6.4. DNSSEC Timing Considerations

As with all DNSSEC signatures, the ability to perform signature validation of a ZONEMD record is limited in time. If the DS record(s) or trust anchors for the zone to be verified are no longer available, the recipient cannot validate the ZONEMD RRSets. This could happen even if the ZONEMD signature is still current (not expired), since the zone's DS record(s) may have been withdrawn following a Key Signing Key (KSK) rollover.

For zones where it may be important to validate a ZONEMD RRSes through its entire signature validity period, the zone operator should ensure that KSK rollover timing takes this into consideration.

6.5. Attacks Utilizing ZONEMD Queries

Nothing in this specification prevents clients from making, and servers from responding to, ZONEMD queries. Servers SHOULD NOT calculate zone digests dynamically (for each query) as this can be used as a CPU resource exhaustion attack.

ZONEMD responses could be used in a distributed denial-of-service amplification attack. The ZONEMD RR is moderately sized, much like the DS RR. A single ZONEMD RR contributes approximately 65 to 95 octets to a DNS response, for digest types defined herein. Other RR types, such as DNSKEY, can result in larger amplification effects.

6.6. Resilience and Fragility

ZONEMD is used to detect incomplete or corrupted zone data prior to its use, thereby increasing resilience by not using corrupt data, but also introduces some denial-of-service fragility by making good data in a zone unavailable if some other data is missing or corrupt. Publishers and consumers of zones containing ZONEMD records should be aware of these tradeoffs. While the intention is to secure the zone data, misconfigurations or implementation bugs are generally indistinguishable from intentional tampering, and could lead to service failures when verification is performed automatically.

Zone publishers may want to deploy ZONEMD gradually, perhaps by utilizing one of the private use hash algorithm code points listed in Section 5.3. Similarly, recipients may want to initially configure verification failures only as a warning, and later as an error after gaining experience and confidence with the feature.

7. Performance Considerations

This section is provided to make zone publishers aware of the performance requirements and implications of including ZONEMD RRs in a zone.

7.1. SIMPLE SHA384

As mentioned previously, the SIMPLE scheme may be impractical for use in zones that are either large or highly dynamic. Zone publishers should carefully consider the use of ZONEMD in such zones, since it might cause consumers of zone data (e.g., secondary name servers) to expend resources on digest calculation. For such use cases, it is recommended that ZONEMD only be used when digest calculation time is significantly less than propagation times and update intervals.

The authors' implementation (Appendix B.1) includes an option to record and report CPU usage of its operation. The software was used to generate digests for more than 800 TLD zones available from [CZDS]. The table below summarizes the results for the SIMPLE scheme and SHA384 hash algorithm grouped by zone size. The Rate column is the mean amount of time per RR to calculate the digest, running on commodity hardware in early 2020.

Zone Size (RRs)	Rate (msec/RR)
10 - 99	0.00683
100 - 999	0.00551
1000 - 9999	0.00505
10000 - 99999	0.00602
100000 - 999999	0.00845
1000000 - 9999999	0.0108
10000000 - 99999999	0.0148

For example, based on the above table, it takes approximately 0.13 seconds to calculate a SIMPLE SHA384 digest for a zone with 22,000 RRs, and about 2.5 seconds for a zone with 300,000 RRs.

These benchmarks attempt to emulate a worst-case scenario and take into account the time required to canonicalize the zone for processing. Each of the 800+ zones were measured three times, and then averaged, with a different random sorting of the input data prior to each measurement.

8. Privacy Considerations

This specification has no impact on user privacy.

9. Acknowledgments

The authors wish to thank David Blacka, Scott Hollenbeck, and Rick Wilhelm for providing feedback on early drafts of this document. Additionally, they thank Joe Abley, Mark Andrews, Ralph Dolmans, Donald Eastlake, Richard Gibson, Olafur Gudmundsson, Bob Harold, Paul Hoffman, Evan Hunt, Shumon Huque, Tatuya Jinmei, Mike St. Johns, Burt Kaliski, Shane Kerr, Matt Larson, Barry Leiba, John Levine, Ed Lewis, Matt Pounsett, Mukund Sivaraman, Petr Spacek, Ondrej Sury, Willem Toorop, Florian Weimer, Tim Wicinski, Wouter Wijngaards, Paul Wouters, and other members of the DNSOP working group for their input.

10. Change Log

RFC Editor: Please remove this section before publication.

This section lists substantial changes to the document as it is being worked on.

From -00 to -01:

- o Removed requirement to sort by RR CLASS.
- o Added Kumari and Hardaker as coauthors.
- o Added Change Log section.
- o Minor clarifications and grammatical edits.

From -01 to -02:

- o Emphasize desire for data security over channel security.
- o Expanded motivation into its own subsection.
- o Removed discussion topic whether or not to include serial in ZONEMD.
- o Clarified that a zone's NS records always sort before the SOA record.
- o Clarified that all records in the zone must be digested, except as specified in the exclusion rules.
- o Added for discussion out-of-zone and occluded records.
- o Clarified that update of ZONEMD signature must not cause a serial number change.
- o Added persons to acknowledgments.

From -02 to -03:

- o Added recommendation to set ZONEMD TTL to SOA TTL.
- o Clarified that digest input uses uncompressed names.
- o Updated Implementations section.
- o Changed intended status from Standards Track to Experimental and added Scope of Experiment section.
- o Updated Motivation, Introduction, and Design Overview sections in response to working group discussion.
- o Gave ZONEMD digest types their own status, separate from DS digest types. Request IANA to create a registry.
- o Added Reserved field for future work supporting dynamic updates.

- o Be more rigorous about having just ONE ZONEMD record in the zone.
- o Expanded use cases.

From -03 to -04:

- o Added an appendix with example zones and digests.
- o Clarified that only apex ZONEMD RRs shall be processed.

From -04 to -05:

- o Made SHA384 the only supported ZONEMD digest type.
- o Disassociated ZONEMD digest types from DS digest types.
- o Updates to Introduction based on list feedback.
- o Changed "zone file" to "zone" everywhere.
- o Restored text about why ZONEMD has a Serial field.
- o Clarified ordering of RRSets having same owner to be numerically ascending.
- o Clarified that all duplicate RRs (not just SOA) must be suppressed in digest calculation.
- o Clarified that the Reserved field must be set to zero and checked for zero in verification.
- o Clarified that occluded data must be included.
- o Clarified procedure for verification, using temporary location for received digest.
- o Explained why Reserved field is 8-bits.
- o IANA Considerations section now more specific.
- o Added complex zone to examples.
- o

From -05 to -06:

- o RR type code 63 was assigned to ZONEMD by IANA.

From -06 to -07:

- o Fixed mistakes in ZONEMD examples.
- o Added private use Digest Type values 240-254.
- o Clarified that Digest field must not be empty.

From -07 to draft-ietf-dnsop-dns-zone-digest-00:

- o Adopted by dnsop.
- o Clarified further that non-apex ZONEMD RRs have no meaning.
- o Changed "provably [un]signed" to "provably [in]secure".
- o Allow multiple ZONEMD RRs to support algorithm agility/rollovers.
- o Describe verification when there are multiple ZONEMD RRs.

From -00 to -01:

- o Simplified requirements around verifying multiple digests. Any one match is sufficient.
- o Updated implementation notes.
- o Both implementations produce expected results on examples given in this document.

From -01 to -02:

- o Changed the name of the Reserved field to Parameter.
- o Changed the name of Digest Type 1 from SHA384 to SHA384-STABLE.
- o The meaning of the Parameter field now depends on Digest Type.
- o No longer require Parameter field to be zero in verification.
- o Updated a rule from earlier versions that said multiple ZONEMD RRs were not allowed.

From -02 to -03:

- o Changed the name of Digest Type 1 from SHA384-STABLE to SHA384-SIMPLE.

- o Changed document status from Experimental to Standards Track.
- o Removed Scope of Experimentation section.

From -03 to -04:

- o Addressing WGLC feedback.
- o Changed from "Digest Type + Parameter" to "Scheme + Hash Algorithm". This should make it more obvious how ZONEMD can be expanded in the future with new schemes and hash algorithms, while sacrificing some of the flexibility that the Parameter was intended to provide.
- o Note: old RDATA fields: Serial, Digest Type, Parameter, Digest.
- o Note: new RDATA fields: Serial, Scheme, Hash Algorithm, Digest.
- o Add new IANA requirement for a Scheme registry.
- o Rearranged some sections and separated scheme-specific aspects from general aspects of digest calculation.
- o When discussing multiple ZONEMD RRs, allow for Scheme, as well as Hash Algorithm, transition.
- o Added Performance Considerations section with some benchmarks.
- o Further clarifications about non-apex ZONEMD RRs.
- o Clarified inclusion rule for duplicate RRs.
- o Removed or lowercased some inappropriately used RFC 2119 key words.
- o Clarified that all ZONEMD RRs, even for unsupported hash algorithms, must be zeroized during digest calculation.
- o Added Resilience and Fragility to security considerations.
- o Updated examples since changes in this version result in different hash values.

From -04 to -05:

- o Clarifications about non-apex and multiple ZONEMD RRs.
- o Clarifications about benchmark results.

- o Don't compute ZONEMD on-the-fly.
- o Specification Required for updates to ZONEMD protocol registries.
- o Other rewording based on WGLC feedback.
- o Updated RFC numbers for some references.
- o Use documentation IP addresses instead of loopback.
- o Updated examples in the appendix.

From -05 to -06:

- o Per WG suggestion, no longer include any apex ZONEMD record in digest calculation.
- o Updated examples in the appendix.
- o Clarified verification procedure by describing a loop over all ZONEMD RRs.

From -06 to -07:

- o Added NIC Chile Labs implementation.

From -07 to -08:

- o Update an author's affiliation.
- o Clarified why placeholder RRs are still important (for NSEC/NSEC3).
- o Moved subsection ("Order of RRsets Having the Same Owner Name") with single sentence paragraph up into parent section.

From -08 to -09:

- o Moved format, ordering, inclusion/exclusion into a sub section specific to the SIMPLE scheme.
- o Further clarified rules about multiple ZONEMD RRs (AD comments).
- o Reworded rules about processing of duplicate zone RRs (AD comments).
- o Removed sentence about optional zeroing of digest prior to calculation (AD comments).

- o Other minor changes (AD comments).

From -09 to -10:

- o Add clarification and reference to on-disk modification / corruption of zone files.
- o Added concerns that timing of KSK rollovers could affect validation of ZONEMD record.
- o Addressed SECDIR review and accepted most proposed edits.
- o From SECDIR review, require minimum digest length of 12 octets.
- o From SECDIR review, add SHA512 has hash algorithm 2.
- o From SECDIR review, say that ZONEMD RRs MAY be ignored by local policy.
- o Moved Implementation Status to an appendix with the intention to retain it in RFC.
- o In registry tables, changed Status column to Implementation Requirement.

From -10 to -11:

- o Fixed people's names in the acknowledgments section (blush)
- o Say "has not been modified between origination and retrieval."
- o Say that ZONEMD TTL doesn't matter during verification.
- o Further clarification that the SHA-384 and SHA-512 hashes are not truncated. Future algs might be truncated, but never below 96 bits.

From -11 to -12:

- o SECDIR review: make "recommended" all caps.
- o SECDIR review: tweak explanation of why ZONEMD RR has copy of SOA serial.
- o SECDIR review: be even more clear about apex ZONEMD RRs vs non-apex.

- o SECDIR review: Forgot to delete sentence about IANA policy for adding new hash algorithms.
- o SECDIR review: Spell out Key Signing Key first time.
- o SECDIR review: say "private use hash algorithm code points."
- o SECDIR review: Update estimates of ZONEMD RR size.

From -12 to -13:

- o Added reference to draft-ietf-dprive-xfr-over-tls.
- o Dropped Implementation Requirement from registry tables.
- o Added Use of Multiple ZONEMD Hash Algorithms to Security Considerations.
- o Added Using Zone Digest Without DNSSEC to Security Considerations.
- o Added notes about the need for algorithm agility due to crypto algorithm aging.
- o Further clarified that only with DNSSEC can ZONEMD guarantee integrity and authenticity.
- o For unsigned zones, ZONEMD serves only as a checksum.
- o Calculation algorithm is designed for common case of offline signing. Deviations may be allowed as long as the end result is the same.
- o Numerous small edits and clarifications from IESG reviewer comments.

From -13 to -14:

- o A few more edits and clarifications from IESG reviewer comments.
- o Moved paragraph about multiple digests to new section titled Including ZONEMD RRs in a Zone.
- o MUST be implemented -> MUST be supported by implementations.
- o Consolidated SHOULD requirements about error reporting to single place at the conclusion of verification.

- o Rephrased "provably insecure" etc as using DNSSEC validation to know whether or not the zone is expected to have keys and signatures.

11. References

11.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://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, <<https://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, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [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/info/rfc8174>>.

11.2. Informative References

- [CZDS] Internet Corporation for Assigned Names and Numbers, "Centralized Zone Data Service", October 2018, <<https://czds.icann.org/>>.
- [disk-full-failure] DENIC, "Background of the Partial Failure of the Name Service for .de Domains", May 2010, <<https://web.archive.org/web/20100618032705/https://www.denic.de/en/denic-in-dialogue/news/2733.html>>.

[DnsTools]

NIC Chile Labs, "DNS tools for zone signature (file, pkcs11-hsm) and validation, and zone digest (ZONEMD)", April 2020, <<https://github.com/niclabs/dns-tools>>.

[I-D.ietf-dprive-xfr-over-tls]

Toorop, W., Dickinson, S., Sahib, S., Aras, P., and A. Mankin, "DNS Zone Transfer-over-TLS", draft-ietf-dprive-xfr-over-tls-02 (work in progress), July 2020.

[InterNIC]

ICANN, "InterNIC FTP site", May 2018, <<ftp://ftp.internic.net/domain/>>.

[ldns-zone-digest]

Verisign, "Implementation of Message Digests for DNS Zones using the ldns library", July 2018, <<https://github.com/verisign/ldns-zone-digest>>.

[RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, DOI 10.17487/RFC1995, August 1996, <<https://www.rfc-editor.org/info/rfc1995>>.

[RFC2065] Eastlake 3rd, D. and C. Kaufman, "Domain Name System Security Extensions", RFC 2065, DOI 10.17487/RFC2065, January 1997, <<https://www.rfc-editor.org/info/rfc2065>>.

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<https://www.rfc-editor.org/info/rfc2136>>.

[RFC2535] Eastlake 3rd, D., "Domain Name System Security Extensions", RFC 2535, DOI 10.17487/RFC2535, March 1999, <<https://www.rfc-editor.org/info/rfc2535>>.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <<https://www.rfc-editor.org/info/rfc2845>>.

[RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <<https://www.rfc-editor.org/info/rfc2931>>.

[RFC3258] Hardie, T., "Distributing Authoritative Name Servers via Shared Unicast Addresses", RFC 3258, DOI 10.17487/RFC3258, April 2002, <<https://www.rfc-editor.org/info/rfc3258>>.

- [RFC4880] Callas, J., Donnerhackle, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <<https://www.rfc-editor.org/info/rfc4880>>.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <<https://www.rfc-editor.org/info/rfc5155>>.
- [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <<https://www.rfc-editor.org/info/rfc5751>>.
- [RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <<https://www.rfc-editor.org/info/rfc5936>>.
- [RFC7696] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015, <<https://www.rfc-editor.org/info/rfc7696>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [RFC8806] Kumari, W. and P. Hoffman, "Running a Root Server Local to a Resolver", RFC 8806, DOI 10.17487/RFC8806, June 2020, <<https://www.rfc-editor.org/info/rfc8806>>.
- [RFC8901] Huque, S., Aras, P., Dickinson, J., Vcelak, J., and D. Blacka, "Multi-Signer DNSSEC Models", RFC 8901, DOI 10.17487/RFC8901, September 2020, <<https://www.rfc-editor.org/info/rfc8901>>.

[RootServers]

Root Server Operators, "Root Server Technical Operations",
July 2018, <<https://www.root-servers.org/>>.

[RPZ]

Wikipedia, "Response policy zone", May 2020,
<https://en.wikipedia.org/w/index.php?title=Response_policy_zone&oldid=960043728>.

[ZoneDigestHackathon]

Kerr, S., "Prototype implementation of ZONEMD for the IETF
102 hackathon in Python", July 2018,
<<https://github.com/shane-kerr/ZoneDigestHackathon>>.

Appendix A. Example Zones With Digests

This appendix contains example zones with accurate ZONEMD records.
These can be used to verify an implementation of the zone digest
protocol.

A.1. Simple EXAMPLE Zone

Here, the EXAMPLE zone contains an SOA record, NS and glue records,
and a ZONEMD record.

```
example.      86400   IN   SOA      ns1 admin 2018031900 (
                    1800 900 604800 86400 )
                    86400   IN   NS       ns1
                    86400   IN   NS       ns2
                    86400   IN   ZONEMD   2018031900 1 1 (
                                                c68090d90a7aed71
                                                6bc459f9340e3d7c
                                                1370d4d24b7e2fc3
                                                a1ddc0b9a87153b9
                                                a9713b3c9ae5cc27
                                                777f98b8e730044c )
ns1            3600   IN   A         203.0.113.63
ns2            3600   IN   AAAA      2001:db8::63
```

A.2. Complex EXAMPLE Zone

Here, the EXAMPLE zone contains duplicate RRs, and an occluded RR,
and one out-of-zone RR.

```

example.      86400   IN   SOA      ns1 admin 2018031900 (
                1800 900 604800 86400 )
                86400   IN   NS       ns1
                86400   IN   NS       ns2
                86400   IN   ZONEMD    2018031900 1 1 (
                                   31cefb03814f5062
                                   ad12fa951ba0ef5f
                                   8da6ae354a415767
                                   246f7dc932ceb1e7
                                   42a2108f529db6a3
                                   3a11c01493de358d )
ns1            3600    IN   A         203.0.113.63
ns2            3600    IN   AAAA      2001:db8::63
occluded.sub   7200    IN   TXT      "I'm occluded but must be digested"
sub            7200    IN   NS       ns1
duplicate      300     IN   TXT      "I must be digested just once"
duplicate      300     IN   TXT      "I must be digested just once"
foo.test.      555     IN   TXT      "out-of-zone data must be excluded"
non-apex       900     IN   ZONEMD    2018031900 1 1 (
                                   616c6c6f77656420
                                   6275742069676e6f
                                   7265642e20616c6c
                                   6f77656420627574
                                   2069676e6f726564
                                   2e20616c6c6f7765 )

```

A.3. EXAMPLE Zone with multiple digests

Here, the EXAMPLE zone contains multiple ZONEMD records. It has both SHA384 and SHA512 digests using the SIMPLE scheme. It also includes ZONEMD records with Scheme and Hash Algorithm values in the private range (240-254). These additional private-range digests are not verifiable.

```

example.      86400    IN    SOA      ns1 admin 2018031900 (
                  1800 900 604800 86400 )
example.      86400    IN    NS       ns1.example.
example.      86400    IN    NS       ns2.example.
example.      86400    IN    ZONEMD   2018031900 1 1 (
                  62e6cf51b02e54b9
                  b5f967d547ce4313
                  6792901f9f88e637
                  493daaf401c92c27
                  9dd10f0edb1c56f8
                  080211f8480ee306 )
example.      86400    IN    ZONEMD   2018031900 1 2 (
                  08cfa1115c7b948c
                  4163a901270395ea
                  226a930cd2cbcf2f
                  a9a5e6eb85f37c8a
                  4e114d884e66f176
                  eab121cb02db7d65
                  2e0cc4827e7a3204
                  f166b47e5613fd27 )
example.      86400    IN    ZONEMD   2018031900 1 240 (
                  e2d523f654b9422a
                  96c5a8f44607bbee )
example.      86400    IN    ZONEMD   2018031900 241 1 (
                  e1846540e33a9e41
                  89792d18d5d131f6
                  05fc283e )
ns1.example.  3600     IN    A        203.0.113.63
ns2.example.  86400    IN    TXT      "This example has multiple digests"
ns2.example.  3600     IN    AAAA     2001:db8::63

```

A.4. The URI.ARPA Zone

The URI.ARPA zone retrieved 2018-10-21. Note this sample zone has (expired) signatures, but no signature for the ZONEMD RR.

```

; <<>> DiG 9.9.4 <<>> @lax.xfr.dns.icann.org uri.arpa axfr
; (2 servers found)
;; global options: +cmd
uri.arpa.      3600     IN    SOA      sns.dns.icann.org. (
                  noc.dns.icann.org. 2018100702 10800 3600 1209600 3600 )
uri.arpa.      3600     IN    RRSIG     NSEC 8 2 3600 (
                  20181028142623 20181007205525 47155 uri.arpa.
                  eEC4w/oXLR1Epwgv4MBiDtSBsXhqrJVvJWUpbX8XpetAvD35bxwNCUTi
                  /pAJVUXefegWeiriD2rkTgCBCMmn7YQIm3gdR+HjY/+o3BXNQnz97f+e
                  HAE9EDDzoNVfLL1PyV/2fde9tDeUuAGVVwmD399NGq9jWYMRpyri2kysr q/g= )
uri.arpa.      86400    IN    RRSIG     NS 8 2 86400 (
                  20181028172020 20181007175821 47155 uri.arpa.

```

```

ATyV2A2A8ZoggC+68u4GuP5MOUuR+2rr3eWOkEU55zAHld/7FiBx14ln
4byJYy7NudUw1MOEXajqFZE7DV18PpcvrP3HeeGaVzKqaWj+aus0jbKF
Bsvs2b1qDZemBfkz/IfAhUTJKnto0vSUicJKfItu0GjyYNJCz2CqEuGD Wxc= )
uri.arpa.          600      IN          RRSIG      MX 8 2 600 (
20181028170556 20181007175821 47155 uri.arpa.
e7/r3KXDohX1lyVavetFFObp8fB8aXT76HnN9KCQDxSnSghNM83UQV0t
lTtD8JVeN1mCvcNFZpagwIgB7XhTtm6Beur/m5ES+4uSnVeS6Q66HBZK
A3mR95IpevuVIZvvJ+GcCAQpBo6KRODYvJ/c/ZG6sfYWkZ7qg/Em5/+3 4UI= )
uri.arpa.          3600     IN          RRSIG      DNSKEY 8 2 3600 (
20181028152832 20181007175821 15796 uri.arpa.
nzbpbnh0OqsgBBP8St28pLvPEQ3wZAUdEBuUwil+rtjjWlYYiqjPxZ286
XF4RqlusfV5x71jZz5IqswOaQgia91ylodFpLuXD6FTGs2nXGhNKKg1V
chHgtwj70mXU72GefVgo8TxrFYzxeEFP5ZTP92t97FVWVvyF86sbbR
6DZj3uA2wEvqBVLEcGJLrMQ9Yy7MueJL3UA4h4E6z02JY9Yp0W9woq0B
dqkkwYTwzogyYfffPmGAJG91RJ2h6cHtFjEze2MnaY2glqniZ0WT9vXXd
uFPm0KD9U77Ac+ZtctAF9tsZwSdAoL365E2LlusZbA+K0BnPPqGFJRJK
5R0A1w== )
uri.arpa.          3600     IN          RRSIG      DNSKEY 8 2 3600 (
20181028152832 20181007175821 55480 uri.arpa.
lWtQV/5szQjkXmbcD47/+rOW8kJPksRFHlzxmxzt906+DBYyfrH6uq5X
nHvrU1QO6M12uhqDeL+bDFVgqSpNy+42/OaZvaK3J8EzPZVBHPJykKMV
63T83aAiJrAyHzOaEdmzLCpalqcEE2ImzlLHSafManRfJL8Yuv+JDZFj
2WDWFecUuwkmIZWX11zxp+DxwzyU1Rl7x4+ok5iKZWig5UnBAf6B8T75
WnXzlhCw3F2pXI0a5LYg71L3Tp/xhjN6Yy9jG1IRf5BjB59X2zra3a2R
PkI09SSnuEwHyF1mDaV5BmQrLGRnCjvwXA7ho2m+vv4SP5dUdXf+GTaA
1HeBfw== )
uri.arpa.          3600     IN          RRSIG      SOA 8 2 3600 (
20181029114753 20181008222815 47155 uri.arpa.
qn8yBNoHDjGdT79U2Wu9IIahoS0YPOgYP8lG+qwPcrZ1BwGiHywuoUa2
Mx6BWZlg+HDyaxj2iOmx+IIqoUHhXUbO7IUkJf1grOKCgAR2twDHRXu
9BUQHy9SoV16wYm3kBTepYxW5FFm8vcdnKAF7sxSY8BbaYNpRIEjDx4A JUC= )
uri.arpa.          3600     IN          NSEC       ftp.uri.arpa. NS SOA (
MX RRSIG NSEC DNSKEY )
uri.arpa.          86400    IN          NS          a.iana-servers.net.
uri.arpa.          86400    IN          NS          b.iana-servers.net.
uri.arpa.          86400    IN          NS          c.iana-servers.net.
uri.arpa.          86400    IN          NS          ns2.lacnic.net.
uri.arpa.          86400    IN          NS          sec3.apnic.net.
uri.arpa.          600      IN          MX          10 pechora.icann.org.
uri.arpa.          3600     IN          DNSKEY      256 3 8 (
AwEAAcBi7tSart2J599zbYWspMNGN70IBWb4ziqyQYH9MTB/VCz6WyUK
uXunwiJJbbQ3bcLqTLWEw134B6cTMHrZpjTAb5WAwg4XcWUu8mdcPTiL
Bl6qVRlRD0WiFCTzuYUfkwsh1Rbr7rvrxSQhF5rh71zSpwV5jjjp65Wx
SdJjlH0B )
uri.arpa.          3600     IN          DNSKEY      257 3 8 (
AwEAAbNVv6ulgRdO3lMtAehz7j3ALRjwZglWesnzvllQl/+hBRZr9QoY
cO2I+DkO4Q1NKxox4DUIxj8SxPO3GwDuOFR9q2/CFi2O0mZjafbdYtWc
3zSdBbi3q0cwCIx7GuG9eq1L+pg7mdk9dgdNZfHwB0LnqTD8ebLPsrO/

```

```

Id7kBaIqYOfMlZnh2fp+2h6OOJZHtY0DK1UlssyB5PKsE0tVzo5s6zo9
iXKe5u+8WTMaGDY49vG80JPAKE7ezMiH/NZcUMiE0PRZ8D3foq2dYuS5
ym+vA83Z7v8A+Rwh4UGnjxKB8zmr803V0ASAmHz/gwH5Vb0nH+LObwFt
l3wpbpb+Wpm8= )
uri.arpa. 3600 IN DNSKEY 257 3 8 (
AwEAAbwnFTakCvaUKsXji4mgmxZUJi1IygbnGahbkmFEa0Ll6J+TchKR
wcgzVfsxUGa2MmeA4hgkAooC3uy+tTmoMsgy8uq/JAaj24DjiHzd46LfD
FK/qMidVqFpYSHeq2Vv5oJkuIsx4oe4KsafGWYN0czKZgH5loGjN2aJG
mrIm++XCphOskgCsQYl65MIzuXffzJyx1Aut+ecAIiVeqRaqQfr8LRU
7wIsLxinXirprtQrbor+EtvlHp9qXE6ARTZDzf4jvsNpKvLFZtmxzFf3
e/UJz5eHjpwDSiZL7xE8aEloInGfPtJx9ZnB3bapltaj5wY+5XOCKgY0
xmJVvNqlwdE= )
ftp.uri.arpa. 3600 IN RRSIG NSEC 8 3 3600 (
20181028080856 20181007175821 47155 uri.arpa.
HClGAqPxzkYkAT7Q/QntQeB6YrkP6EPOef+9Qo5/2zngwAewXEAQiyF9
jDlUSJiroml1QqBS3v3aIdW/LXORs4Ez3hLcKN0lcKHsOuWAqzmE+BPP
Arfh8N95jqh/q6vpaB9UtMkQ53tM2fYU1GszOLN0knxbHgDHAh2axMGH 1qM= )
ftp.uri.arpa. 604800 IN RRSIG NAPTR 8 3 604800 (
20181028103644 20181007205525 47155 uri.arpa.
WoLi+vZzkxaoLr2IGZnwkrvcDf6KxiWQdlWZP/U+AWnV+7MiqsWPZaf0
9toRErerGoFOiOASNxZjBGJrRgjmavOM9U+LZSconP9zrNFd4dIu6kp5
YxlQJ0uHOvx1ZHFCj6lAt1ACUIw04ZhMydTmi27c8MzEOMepvn7iH7r7 k7k= )
ftp.uri.arpa. 3600 IN NSEC http.uri.arpa. NAPTR (
RRSIG NSEC )
ftp.uri.arpa. 604800 IN NAPTR 0 0 "" "" (
"!^ftp://([^:/?#]*).*$!\\1!i" . )
http.uri.arpa. 3600 IN RRSIG NSEC 8 3 3600 (
20181029010647 20181007175821 47155 uri.arpa.
U03NntQ73LHWpfLmUK8nMsqkwVsOGW2KdsyuHYAjqQSZvKbtmbv7HBmE
H1+Ii3Z+wtfdMZBy5aC/6sHdx69BfZJs16xumycMlAy6325DKTQbIMN+
ift9GrKBC7cgCd2msF/uzSrYxxg4MJQzBPv1kwXnY3b7eJSlIXisBIn7 3b8= )
http.uri.arpa. 604800 IN RRSIG NAPTR 8 3 604800 (
20181029011815 20181007205525 47155 uri.arpa.
T7mRrdag+WSmG+n22mtBSQ/0Y3v+rdDnfQV90LN5Fq32N5K2iYFajF7F
Tp56oOznytfcL4fHrqOE0wRc9NWOCUec9C7WalqJQCllEvgoAM+L6f0
RsEjWq6+9jv1LKMXQv0xQuMX17338uoD/xiAFQSnDbiQKxwWMqVAimv5 7Zs= )
http.uri.arpa. 3600 IN NSEC mailto.uri.arpa. NAPTR (
RRSIG NSEC )
http.uri.arpa. 604800 IN NAPTR 0 0 "" "" (
"!^http://([^:/?#]*).*$!\\1!i" . )
mailto.uri.arpa. 3600 IN RRSIG NSEC 8 3 3600 (
20181028110727 20181007175821 47155 uri.arpa.
GvxzVL85rEukwGgtuLxek9ipwjBMfTOFIEyJ7afC8HxVMs6mfFa/nEM/
IdFvvFg+lcYoJSQYuSAVYF13xPbgrxVSLK125QutCFMdc/YjuZEnq5cl
fQciMRD7R3+znZfm8d8u/snLV9w4D+1TBZrJJUBelEfc8vum5vvV7819 ZoY= )
mailto.uri.arpa. 604800 IN RRSIG NAPTR 8 3 604800 (
20181028141825 20181007205525 47155 uri.arpa.
MaADUgc3fc5v++M0YmqjGk3jBdfIA5RuP62hUSlPsFZO4k37erjIGCfF

```



```

j+g84yc+QgbSde0PQHszl9fE/+SU5ZXiS9YdcbzSZxp2erFpZOTchrg
916T4vx6i59scodjb016bDyZ+mtIPrc1w6b4hUyOUTsDQoAJYxdfEuMg Vy4= )
mailto.uri.arpa. 3600 IN NSEC urn.uri.arpa. NAPTR (
RRSIG NSEC )
mailto.uri.arpa. 604800 IN NAPTR 0 0 "" "" (
"!^mailto:(.*)@(.*)$!\2!i" . )
urn.uri.arpa. 3600 IN RRSIG NSEC 8 3 3600 (
20181028123243 20181007175821 47155 uri.arpa.
Hgsw4Deops108uWyELGe6hpR/OEqCnTHvahlwiQkHhO5CSEQrbhmFAWe
UOkmGAdTEYrSz+skLRQuITRMwzyFf4oUkZihGyhZyzHbcxWfuDc/Pd/9
DSl56gdeBwylevn5wBTms8yWQVKNtphbJH395gRqZuaJs3LD/qTyJ5Dp LvA= )
urn.uri.arpa. 604800 IN RRSIG NAPTR 8 3 604800 (
20181029071816 20181007205525 47155 uri.arpa.
ALIZD0vBqAQQt40GQOEfaj8OCyE9xSRJRdyvyn/H/wZVXFrfKrQYrLAS
D/K7q6CMT0xTRCu2J8yes63WJiaJEdnh+dscXzZkmOg4n5PsgZbkvUSW
BiGtxvz5jNncM0xVbkjbtByrvJQA0lcUlmnlDKelFmVBluLpVdA9Ib4J hMU= )
urn.uri.arpa. 3600 IN NSEC uri.arpa. NAPTR RRSIG (
NSEC )
urn.uri.arpa. 604800 IN NAPTR 0 0 "" "" (
"/urn:([^\:]+)/\1/i" . )
uri.arpa. 3600 IN SOA sns.dns.icann.org. (
noc.dns.icann.org. 2018100702 10800 3600 1209600 3600 )
;; Query time: 66 msec
;; SERVER: 192.0.32.132#53(192.0.32.132)
;; WHEN: Sun Oct 21 20:39:28 UTC 2018
;; XFR size: 34 records (messages 1, bytes 3941)
uri.arpa. 3600 IN ZONEMD 2018100702 1 1 (
1291b78ddf7669b1a39d014d87626b709b55774c5d7d58fa
dc556439889a10eaf6f11d615900a4f996bd46279514e473 )

```

A.5. The ROOT-SERVERS.NET Zone

The ROOT-SERVERS.NET zone retrieved 2018-10-21.

```

root-servers.net.      3600000 IN  SOA      a.root-servers.net. (
    nstld.verisign-grs.com. 2018091100 14400 7200 1209600 3600000 )
root-servers.net.      3600000 IN  NS       a.root-servers.net.
root-servers.net.      3600000 IN  NS       b.root-servers.net.
root-servers.net.      3600000 IN  NS       c.root-servers.net.
root-servers.net.      3600000 IN  NS       d.root-servers.net.
root-servers.net.      3600000 IN  NS       e.root-servers.net.
root-servers.net.      3600000 IN  NS       f.root-servers.net.
root-servers.net.      3600000 IN  NS       g.root-servers.net.
root-servers.net.      3600000 IN  NS       h.root-servers.net.
root-servers.net.      3600000 IN  NS       i.root-servers.net.
root-servers.net.      3600000 IN  NS       j.root-servers.net.
root-servers.net.      3600000 IN  NS       k.root-servers.net.
root-servers.net.      3600000 IN  NS       l.root-servers.net.
root-servers.net.      3600000 IN  NS       m.root-servers.net.
a.root-servers.net.    3600000 IN  AAAA     2001:503:ba3e::2:30
a.root-servers.net.    3600000 IN  A        198.41.0.4
b.root-servers.net.    3600000 IN  MX       20 mail.isi.edu.
b.root-servers.net.    3600000 IN  AAAA     2001:500:200::b
b.root-servers.net.    3600000 IN  A        199.9.14.201
c.root-servers.net.    3600000 IN  AAAA     2001:500:2::c
c.root-servers.net.    3600000 IN  A        192.33.4.12
d.root-servers.net.    3600000 IN  AAAA     2001:500:2d::d
d.root-servers.net.    3600000 IN  A        199.7.91.13
e.root-servers.net.    3600000 IN  AAAA     2001:500:a8::e
e.root-servers.net.    3600000 IN  A        192.203.230.10
f.root-servers.net.    3600000 IN  AAAA     2001:500:2f::f
f.root-servers.net.    3600000 IN  A        192.5.5.241
g.root-servers.net.    3600000 IN  AAAA     2001:500:12::d0d
g.root-servers.net.    3600000 IN  A        192.112.36.4
h.root-servers.net.    3600000 IN  AAAA     2001:500:1::53
h.root-servers.net.    3600000 IN  A        198.97.190.53
i.root-servers.net.    3600000 IN  MX       10 mx.i.root-servers.org.
i.root-servers.net.    3600000 IN  AAAA     2001:7fe::53
i.root-servers.net.    3600000 IN  A        192.36.148.17
j.root-servers.net.    3600000 IN  AAAA     2001:503:c27::2:30
j.root-servers.net.    3600000 IN  A        192.58.128.30
k.root-servers.net.    3600000 IN  AAAA     2001:7fd::1
k.root-servers.net.    3600000 IN  A        193.0.14.129
l.root-servers.net.    3600000 IN  AAAA     2001:500:9f::42
l.root-servers.net.    3600000 IN  A        199.7.83.42
m.root-servers.net.    3600000 IN  AAAA     2001:dc3::35
m.root-servers.net.    3600000 IN  A        202.12.27.33
root-servers.net.      3600000 IN  SOA      a.root-servers.net. (
    nstld.verisign-grs.com. 2018091100 14400 7200 1209600 3600000 )
root-servers.net.      3600000 IN  ZONEMD   2018091100 1 1 (
    flca0ccd91bd5573d9f431c00ee0101b2545c97602be0a97
    8a3b11dbfclc776d5b3e86ae3d973d6b5349ba7f04340f79 )

```

Appendix B. Implementation Status

RFC Editor: Please retain this section upon publication.

This section records the status of known implementations of the protocol defined by this specification at the time of publication, and is inspired by the concepts described in RFC7942.

Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

B.1. Authors' Implementation

The authors have an open source implementation in C, using the ldns library [ldns-zone-digest]. This implementation is able to perform the following functions:

- o Read an input zone and output a zone with the ZONEMD placeholder.
- o Compute zone digest over signed zone and update the ZONEMD record.
- o Re-compute DNSSEC signature over the ZONEMD record.
- o Verify the zone digest from an input zone.

This implementation does not:

- o Perform DNSSEC validation of the ZONEMD record during verification.

B.2. Shane Kerr's Implementation

Shane Kerr wrote an implementation of this specification during the IETF 102 hackathon [ZoneDigestHackathon]. This implementation is in Python and is able to perform the following functions:

- o Read an input zone and output a zone with ZONEMD record.
- o Verify the zone digest from an input zone.
- o Output the ZONEMD record in its defined presentation format.

This implementation does not:

- o Re-compute DNSSEC signature over the ZONEMD record.
- o Perform DNSSEC validation of the ZONEMD record.

B.3. NIC Chile Labs Implementation

NIC Chile Labs wrote an implementation of this specification as part of "dns-tools" suite [DnsTools], which besides digesting, can also sign and verify zones. This implementation is in Go and is able to perform the following functions:

- o Compute zone digest over signed zone and update the ZONEMD record.
- o Verify the zone digest from an input zone.
- o Perform DNSSEC validation of the ZONEMD record during verification.
- o Re-compute DNSSEC signature over the ZONEMD record.

Authors' Addresses

Duane Wessels
Verisign
12061 Bluemont Way
Reston, VA 20190

Phone: +1 703 948-3200
Email: dwessels@verisign.com
URI: <https://verisign.com>

Piet Barber
Verisign
12061 Bluemont Way
Reston, VA 20190

Phone: +1 703 948-3200
Email: pbarber@verisign.com
URI: <https://verisign.com>

Matt Weinberg
Amazon

Email: matweinb@amazon.com
URI: <https://amazon.com>

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043

Email: warren@kumari.net

Wes Hardaker
USC/ISI
P.O. Box 382
Davis, CA 95617

Email: ietf@hardakers.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 6, 2020

W. Kumari
Google
E. Hunt
ISC
R. Arends
ICANN
W. Hardaker
USC/ISI
D. Lawrence
Oracle + Dyn
May 05, 2020

Extended DNS Errors
draft-ietf-dnsop-extended-error-16

Abstract

This document defines an extensible method to return additional information about the cause of DNS errors. Though created primarily to extend SERVFAIL to provide additional information about the cause of DNS and DNSSEC failures, the Extended DNS Errors option defined in this document allows all response types to contain extended error information. Extended DNS Error information does not change the processing of RCODEs.

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 November 6, 2020.

Copyright Notice

Copyright (c) 2020 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 (<https://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. Introduction and background	3
1.1. Requirements notation	3
2. Extended DNS Error EDNS0 option format	4
3. Extended DNS Error Processing	5
4. Defined Extended DNS Errors	5
4.1. Extended DNS Error Code 0 - Other	6
4.2. Extended DNS Error Code 1 - Unsupported DNSKEY Algorithm	6
4.3. Extended DNS Error Code 2 - Unsupported DS Digest Type	6
4.4. Extended DNS Error Code 3 - Stale Answer	6
4.5. Extended DNS Error Code 4 - Forged Answer	6
4.6. Extended DNS Error Code 5 - DNSSEC Indeterminate	6
4.7. Extended DNS Error Code 6 - DNSSEC Bogus	6
4.8. Extended DNS Error Code 7 - Signature Expired	6
4.9. Extended DNS Error Code 8 - Signature Not Yet Valid	7
4.10. Extended DNS Error Code 9 - DNSKEY Missing	7
4.11. Extended DNS Error Code 10 - RRSIGs Missing	7
4.12. Extended DNS Error Code 11 - No Zone Key Bit Set	7
4.13. Extended DNS Error Code 12 - NSEC Missing	7
4.14. Extended DNS Error Code 13 - Cached Error	7
4.15. Extended DNS Error Code 14 - Not Ready	7
4.16. Extended DNS Error Code 15 - Blocked	7
4.17. Extended DNS Error Code 16 - Censored	7
4.18. Extended DNS Error Code 17 - Filtered	8
4.19. Extended DNS Error Code 18 - Prohibited	8
4.20. Extended DNS Error Code 19 - Stale NXDOMAIN Answer	8
4.21. Extended DNS Error Code 20 - Not Authoritative	8
4.22. Extended DNS Error Code 21 - Not Supported	8
4.23. Extended DNS Error Code 22 - No Reachable Authority	8
4.24. Extended DNS Error Code 23 - Network Error	8
4.25. Extended DNS Error Code 24 - Invalid Data	9
5. IANA Considerations	9
5.1. A New Extended DNS Error Code EDNS Option	9
5.2. New Registry for Extended DNS Error Codes	9
6. Security Considerations	12

7. Acknowledgements	12
8. References	13
8.1. Normative References	13
8.2. Informative References	13
Authors' Addresses	14

1. Introduction and background

There are many reasons that a DNS query may fail, some of them transient, some permanent; some can be resolved by querying another server, some are likely best handled by stopping resolution. Unfortunately, the error signals that a DNS server can return are very limited, and are not very expressive. This means that applications and resolvers often have to "guess" at what the issue is - e.g. was the answer marked REFUSED because of a lame delegation, or because the nameserver is still starting up and loading zones? Is a SERVFAIL a DNSSEC validation issue, or is the nameserver experiencing some other failure? What error messages should be presented to the user or logged under these conditions?

A good example of issues that would benefit from additional error information are errors caused by DNSSEC validation issues. When a stub resolver queries a name which is DNSSEC bogus [RFC8499] (using a validating resolver), the stub resolver receives only a SERVFAIL in response. Unfortunately, the SERVFAIL Response Code (RCODE) is used to signal many sorts of DNS errors, and so the stub resolver's only option is to ask the next configured DNS resolver. The result of trying the next resolver is one of two outcomes: either the next resolver also validates, and a SERVFAIL is returned again, or the next resolver is not a validating resolver, and the user is returned a potentially harmful result. With an Extended DNS Error (EDE) option enclosed in the response message, the resolver is able to return a more descriptive reason as to why any failures happened, or add additional context to a message containing a NOERROR RCODE.

This document specifies a mechanism to extend DNS errors to provide additional information about the cause of an error. These extended DNS error codes are described in this document can be used by any system that sends DNS queries and receives a response containing an EDE option. Different codes are useful in different circumstances, and thus different systems (stub resolvers, recursive resolvers, and authoritative resolvers) might receive and use them.

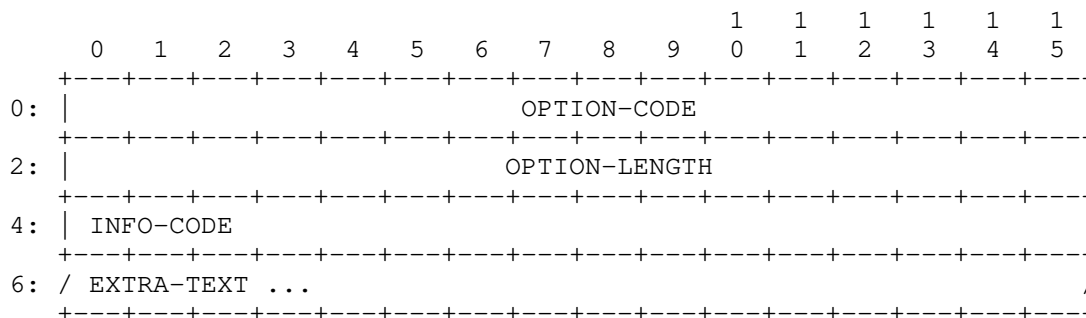
1.1. Requirements notation

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. Extended DNS Error EDNS0 option format

This draft uses an EDNS0 ([RFC6891]) option to include Extended DNS Error (EDE) information in DNS messages. The option is structured as follows:



Field definition details:

- o OPTION-CODE, 2-octets/16-bits (defined in [RFC6891]), for EDE is TBD. [RFC Editor: change TBD to the proper code once assigned by IANA.]
- o OPTION-LENGTH, 2-octets/16-bits ((defined in [RFC6891])) contains the length of the payload (everything after OPTION-LENGTH) in octets and should be 2 plus the length of the EXTRA-TEXT field (which may be a zero-length string).
- o INFO-CODE, 16-bits, which is the principal contribution of this document. This 16-bit value, encoded in network (MSB) byte order, provides the additional context for the RESPONSE-CODE of the DNS message. The INFO-CODE serves as an index into the "Extended DNS Errors" registry defined and created in Section 5.2.
- o EXTRA-TEXT, a variable length, UTF-8 encoded [RFC5198], text field that may hold additional textual information. This information is intended for human consumption (not automated parsing). EDE text may be null terminated but MUST NOT be assumed to be; the length MUST be derived from the OPTION-LENGTH field. The EXTRA-TEXT field may be zero octets in length, indicating that there is no EXTRA-TEXT included. Care should be taken not to include private information in the EXTRA-TEXT field that an observer would not otherwise have access to, such as account numbers.

The Extended DNS Error (EDE) option can be included in any response (SERVFAIL, NXDOMAIN, REFUSED, and even NOERROR, etc) to a query that includes OPT Pseudo-RR [RFC6891]. This document includes a set of

initial codepoints, but is extensible via the IANA registry defined and created in Section 5.2.

3. Extended DNS Error Processing

When the response grows beyond the requestor's UDP payload size [RFC6891], servers SHOULD truncate messages by dropping EDE options before dropping other data from packets. Implementations SHOULD set the truncation bit when dropping EDE options. Because long EXTRA-TEXT fields may trigger truncation (which is undesirable given the supplemental nature of EDE) implementers and operators creating EDE options SHOULD avoid lengthy EXTRA-TEXT contents.

When a resolver or forwarder receives an EDE option, whether or not (and how) to pass along EDE information on to their original client is implementation dependent. Implementations MAY choose to not forward information, or they MAY choose to create a new EDE option(s) that conveys the information encoded in the received EDE. When doing so, the source of the error SHOULD be attributed in the EXTRA-TEXT field, since an EDNS0 option received by the original client will appear to have come from the resolver or forwarder sending it.

This document does not allow or prohibit any particular extended error codes and information to be matched with any particular RCODEs. Some combinations of extended error codes and RCODEs may seem nonsensical (such as resolver-specific extended error codes in responses from authoritative servers), so systems interpreting the extended error codes MUST NOT assume that a combination will make sense. Receivers MUST be able to accept EDE codes and EXTRA-TEXT in all messages, including those with a NOERROR RCODE, but need not act on them. Applications MUST continue to follow requirements from applicable specifications on how to process RCODEs no matter what EDE values are also received. Senders MAY include more than one EDE option and receivers MUST be able to accept (but not necessarily process or act on) multiple EDE options in a DNS message.

4. Defined Extended DNS Errors

This document defines some initial EDE codes. The mechanism is intended to be extensible, and additional code-points can be registered in the "Extended DNS Errors" registry Section 5.2. The INFO-CODE from the EDE EDNS option is used to serve as an index into the "Extended DNS Error" IANA registry, the initial values for which are defined in the following sub-sections.

4.1. Extended DNS Error Code 0 - Other

The error in question falls into a category that does not match known extended error codes. Implementations SHOULD include an EXTRA-TEXT value to augment this error code with additional information.

4.2. Extended DNS Error Code 1 - Unsupported DNSKEY Algorithm

The resolver attempted to perform DNSSEC validation, but a DNSKEY RRSET contained only unsupported DNSSEC algorithms.

4.3. Extended DNS Error Code 2 - Unsupported DS Digest Type

The resolver attempted to perform DNSSEC validation, but a DS RRSET contained only unsupported Digest Types.

4.4. Extended DNS Error Code 3 - Stale Answer

The resolver was unable to resolve the answer within its time limits and decided to answer with previously cached data instead of answering with an error. This is typically caused by problems communicating with an authoritative server, possibly as result of a denial of service (DoS) attack against another network. (See also Code 19.)

4.5. Extended DNS Error Code 4 - Forged Answer

For policy reasons (legal obligation, or malware filtering, for instance), an answer was forged. Note that this should be used when an answer is still provided, not when failure codes are returned instead. See Blocked(15), Censored (16), and Filtered (17) for use when returning other response codes.

4.6. Extended DNS Error Code 5 - DNSSEC Indeterminate

The resolver attempted to perform DNSSEC validation, but validation ended in the Indeterminate state [RFC4035].

4.7. Extended DNS Error Code 6 - DNSSEC Bogus

The resolver attempted to perform DNSSEC validation, but validation ended in the Bogus state.

4.8. Extended DNS Error Code 7 - Signature Expired

The resolver attempted to perform DNSSEC validation, but no signatures are presently valid and some (often all) are expired.

4.9. Extended DNS Error Code 8 - Signature Not Yet Valid

The resolver attempted to perform DNSSEC validation, but but no signatures are presently valid and at least some are not yet valid.

4.10. Extended DNS Error Code 9 - DNSKEY Missing

A DS record existed at a parent, but no supported matching DNSKEY record could be found for the child.

4.11. Extended DNS Error Code 10 - RRSIGs Missing

The resolver attempted to perform DNSSEC validation, but no RRSIGs could be found for at least one RRset where RRSIGs were expected.

4.12. Extended DNS Error Code 11 - No Zone Key Bit Set

The resolver attempted to perform DNSSEC validation, but no Zone Key Bit was set in a DNSKEY.

4.13. Extended DNS Error Code 12 - NSEC Missing

The resolver attempted to perform DNSSEC validation, but the requested data was missing and a covering NSEC or NSEC3 was not provided.

4.14. Extended DNS Error Code 13 - Cached Error

The resolver is returning the SERVFAIL RCODE from its cache.

4.15. Extended DNS Error Code 14 - Not Ready

The server is unable to answer the query as it was not fully functional when the query was received.

4.16. Extended DNS Error Code 15 - Blocked

The server is unable to respond to the request because the domain is blacklisted due to an internal security policy imposed by the operator of the server resolving or forwarding the query.

4.17. Extended DNS Error Code 16 - Censored

The server is unable to respond to the request because the domain is blacklisted due to an external requirement imposed by an entity other than the operator of the server resolving or forwarding the query. Note that how the imposed policy is applied is irrelevant (in-band DNS filtering, court order, etc).

4.18. Extended DNS Error Code 17 - Filtered

The server is unable to respond to the request because the domain is blacklisted as requested by the client. Functionally, this amounts to "you requested that we filter domains like this one."

4.19. Extended DNS Error Code 18 - Prohibited

An authoritative server or recursive resolver that receives a query from an "unauthorized" client can annotate its REFUSED message with this code. Examples of "unauthorized" clients are recursive queries from IP addresses outside the network, blacklisted IP addresses, local policy, etc.

4.20. Extended DNS Error Code 19 - Stale NXDOMAIN Answer

The resolver was unable to resolve an answer within its configured time limits and decided to answer with a previously cached NXDOMAIN answer instead of answering with an error. This may be caused, for example, by problems communicating with an authoritative server, possibly as result of a denial of service (DoS) attack against another network. (See also Code 3.)

4.21. Extended DNS Error Code 20 - Not Authoritative

An authoritative server that receives a query with the RD bit clear, or when it is not configured for recursion for a domain for which it is not authoritative SHOULD include this EDE code in the REFUSED response. A resolver that receives a query with the RD bit clear SHOULD include this EDE code in the REFUSED response.

4.22. Extended DNS Error Code 21 - Not Supported

The requested operation or query is not supported.

4.23. Extended DNS Error Code 22 - No Reachable Authority

The resolver could not reach any of the authoritative name servers (or they potentially refused to reply).

4.24. Extended DNS Error Code 23 - Network Error

An unrecoverable error occurred while communicating with another server.

4.25. Extended DNS Error Code 24 - Invalid Data

The authoritative server cannot answer with data for a zone it is otherwise configured to support. Examples of this include its most recent zone being too old, or having expired.

5. IANA Considerations

5.1. A New Extended DNS Error Code EDNS Option

This document defines a new EDNS(0) option, entitled "Extended DNS Error", assigned a value of TBD from the "DNS EDNS0 Option Codes (OPT)" registry [to be removed upon publication:
[<http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-11>]

Value	Name	Status	Reference
TBD	Extended DNS Error	Standard	[This document]

5.2. New Registry for Extended DNS Error Codes

IANA is requested to create and maintain a new registry table called "Extended DNS Error Codes" on the "Domain Name System (DNS) Parameters" web page as follows:

Registry Name: Extended DNS Error Codes

Registration Procedures:

- o 0 - 49151: First come, first served.
- o 49152 - 65535: Private use.

Reference: [this document]

The Extended DNS Error Codes registry is a table with three columns: INFO-CODE, Purpose, and Reference. The initial contents is as below with [this document] added to each reference given.

INFO-CODE: 0
Purpose: Other Error
Reference: Section 4.1

INFO-CODE: 1
Purpose: Unsupported DNSKEY Algorithm
Reference: Section 4.2

INFO-CODE: 2

Purpose: Unsupported DS Digest Type
Reference: Section 4.3

INFO-CODE: 3
Purpose: Stale Answer
Reference: Section 4.4, [RFC8767]

INFO-CODE: 4
Purpose: Forged Answer
Reference: Section 4.5

INFO-CODE: 5
Purpose: DNSSEC Indeterminate
Reference: Section 4.6

INFO-CODE: 6
Purpose: DNSSEC Bogus
Reference: Section 4.7

INFO-CODE: 7
Purpose: Signature Expired
Reference: Section 4.8

INFO-CODE: 8
Purpose: Signature Not Yet Valid
Reference: Section 4.9

INFO-CODE: 9
Purpose: DNSKEY Missing
Reference: Section 4.10

INFO-CODE: 10
Purpose: RRSIGs Missing
Reference: Section 4.11

INFO-CODE: 11
Purpose: No Zone Key Bit Set
Reference: Section 4.12

INFO-CODE: 12
Purpose: NSEC Missing
Reference: Section 4.13

INFO-CODE: 13
Purpose: Cached Error
Reference: Section 4.14

INFO-CODE: 14

Purpose: Not Ready.
Reference: Section 4.15

INFO-CODE: 15
Purpose: Blocked
Reference: Section 4.16

INFO-CODE: 16
Purpose: Censored
Reference: Section 4.17

INFO-CODE: 17
Purpose: Filtered
Reference: Section 4.18

INFO-CODE: 18
Purpose: Prohibited
Reference: Section 4.19

INFO-CODE: 19
Purpose: Stale NXDomain Answer
Reference: Section 4.20

INFO-CODE: 20
Purpose: Not Authoritative
Reference: Section 4.21

INFO-CODE: 21
Purpose: Not Supported
Reference: Section 4.22

INFO-CODE: 22
Purpose: No Reachable Authority
Reference: Section 4.23

INFO-CODE: 23
Purpose: Network Error
Reference: Section 4.24

INFO-CODE: 24
Purpose: Invalid Data
Reference: Section 4.25

INFO-CODE: 25-65535
Purpose: Unassigned
Reference: Section 5.2

6. Security Considerations

Though DNSSEC continues to be deployed, unfortunately a significant number of clients (~11% according to [GeoffValidation]) that receive a SERVFAIL from a validating resolver because of a DNSSEC validation issue will simply ask the next (potentially non-validating) resolver in their list, and thus don't get the protections which DNSSEC should provide.

EDE information is unauthenticated information, unless secured by a form of secured DNS transaction such as [RFC2845], [RFC2931], [RFC8094] or [RFC8484]. An attacker (e.g a MITM or malicious recursive server) could insert an extended error response into untrusted data -- although ideally clients and resolvers would not trust any unauthenticated information. As such, EDE content should be treated only as diagnostic information and MUST NOT alter DNS protocol processing. Until all DNS answers are authenticated via DNSSEC or the other mechanisms mentioned above, there are some tradeoffs. As an example, an attacker who is able to insert the DNSSEC Bogus Extended Error into a DNS message could instead simply reply with a fictitious address (A or AAAA) record. Note that DNS Response Codes (RCODEs) also contain no authentication and can be just as easily manipulated.

By design, EDE potentially exposes additional information DNS resolution processes that may leak information. An example of this is the Prohibited EDE code (18), which may leak the fact that the name is on a blacklist.

7. Acknowledgements

The authors wish to thank Joe Abley, Mark Andrews, Tim April, Vittorio Bertola, Stephane Bortzmeyer, Vladimir Cunat, Ralph Dolmans, Peter DeVries, Peter van Dijk, Mats Dufberg, Donald Eastlake, Bob Harold, Paul Hoffman, Geoff Huston, Shane Kerr, Edward Lewis, Carlos M. Martinez, George Michelson, Eric Orth, Michael Sheldon, Puneet Sood, Petr Spacek, Ondrej Sury, John Todd, Loganaden Velvindron, and Paul Vixie. They also vaguely remember discussing this with a number of people over the years, but have forgotten who all they were -- if you were one of them, and are not listed, please let us know and we'll acknowledge you.

One author also wants to thank the band "Infected Mushroom" for providing a good background soundtrack (and to see if he can get away with this in an RFC!). Another author would like to thank the band "Mushroom Infectors". This was funny at the time we wrote it, but we cannot remember why...

8. References

8.1. Normative References

- [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/info/rfc2119>>.
- [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, <<https://www.rfc-editor.org/info/rfc4035>>.
- [RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <<https://www.rfc-editor.org/info/rfc5198>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/info/rfc6891>>.
- [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/info/rfc8174>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [RFC8767] Lawrence, D., Kumari, W., and P. Sood, "Serving Stale Data to Improve DNS Resiliency", RFC 8767, DOI 10.17487/RFC8767, March 2020, <<https://www.rfc-editor.org/info/rfc8767>>.

8.2. Informative References

- [GeoffValidation] APNIC, G. H., "A quick review of DNSSEC Validation in today's Internet", June 2016, <<http://www.potaroo.net/presentations/2016-06-27-dnssec.pdf>>.
- [RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <<https://www.rfc-editor.org/info/rfc2845>>.

- [RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <<https://www.rfc-editor.org/info/rfc2931>>.
- [RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <<https://www.rfc-editor.org/info/rfc8094>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.

Authors' Addresses

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: warren@kumari.net

Evan Hunt
ISC
950 Charter St
Redwood City, CA 94063
US

Email: each@isc.org

Roy Arends
ICANN

Email: roy.arends@icann.org

Wes Hardaker
USC/ISI
P.O. Box 382
Davis, CA 95617
US

Email: ietf@hardakers.net

David C Lawrence
Oracle + Dyn
150 Dow St
Manchester, NH 03101
US

Email: tale@dd.org

DNSOP Working Group
Internet-Draft
Updates: 7873 (if approved)
Intended status: Standards Track
Expires: 17 July 2021

O. Sury
Internet Systems Consortium
W. Toorop
NLnet Labs
D. Eastlake 3rd
Futurewei Technologies
M. Andrews
Internet Systems Consortium
13 January 2021

Interoperable Domain Name System (DNS) Server Cookies
draft-ietf-dnsop-server-cookies-05

Abstract

DNS Cookies, as specified in [RFC7873], are a lightweight DNS transaction security mechanism that provide limited protection to DNS servers and clients against a variety of amplification denial of service, forgery, or cache poisoning attacks by off-path attackers.

This document updates [RFC7873] with precise directions for creating Server Cookies so that an anycast server set including diverse implementations will interoperate with standard clients, suggestions for constructing Client Cookies in a privacy preserving fashion, and suggestions on how to update a Server Secret. An IANA registry listing the methods and associated pseudo random function suitable for creating DNS Server Cookies is created, with the method described in this document as the first and as of yet only entry.

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 17 July 2021.

Copyright Notice

Copyright (c) 2021 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 (<https://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. Introduction	3
1.1. Terminology and Definitions	3
2. Changes to [RFC7873]	4
3. Constructing a Client Cookie	4
4. Constructing a Server Cookie	5
4.1. The Version Sub-Field	6
4.2. The Reserved Sub-Field	6
4.3. The Timestamp Sub-Field	6
4.4. The Hash Sub-Field	7
5. Updating the Server Secret	8
6. Cookie Algorithms	9
7. IANA Considerations	9
8. Security and Privacy Considerations	10
8.1. Client Cookie construction	10
8.2. Server Cookie construction	11
9. Acknowledgements	12
10. Normative References	12
11. Informative References	13
Appendix A. Test vectors	13
A.1. Learning a new Server Cookie	13
A.2. The same client learning a renewed (fresh) Server Cookie	14
A.3. Another client learning a renewed Server Cookie	15
A.4. IPv6 query with rolled over secret	16
Appendix B. Implementation status	17
Authors' Addresses	18

1. Introduction

DNS Cookies, as specified in [RFC7873], are a lightweight DNS transaction security mechanism that provide limited protection to DNS servers and clients against a variety of denial of service amplification, forgery, or cache poisoning attacks by off-path attackers. This document specifies a means of producing interoperable Cookies so that an anycast server set including diverse implementations can be easily configured to interoperate with standard clients. Also single implementation or non-anycast services can benefit from a well-studied standardized algorithm for which the behavioural and security characteristics are more widely known.

The threats considered for DNS Cookies and the properties of the DNS Security features other than DNS Cookies are discussed in [RFC7873].

In [RFC7873] in Section 6 it is "RECOMMENDED for simplicity that the same Server Secret be used by each DNS server in a set of anycast servers." However, how precisely a Server Cookie is calculated from this Server Secret, is left to the implementation.

This guidance has led to a gallimaufry of DNS Cookie implementations, calculating the Server Cookie in different ways. As a result, DNS Cookies are impractical to deploy on multi-vendor anycast networks, because even when all DNS Software share the same secret, as RECOMMENDED in Section 6 of [RFC7873], the Server Cookie constructed by one implementation cannot generally be validated by another.

There is no need for DNS client (resolver) Cookies to be interoperable across different implementations. Each client need only be able to recognize its own cookies. However, this document does contain recommendations for constructing Client Cookies in a client protecting fashion.

1.1. Terminology 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.

* "IP address" is used herein as a length independent term covering both IPv4 and IPv6 addresses.

2. Changes to [RFC7873]

In its Appendices A.1 and B.1, [RFC7873] provides example "simple" algorithms for computing Client and Server Cookies, respectively. These algorithms MUST NOT be used as the resulting cookies are too weak when evaluated against modern security standards.

In its Appendix B.2, [RFC7873] provides an example "more complex" server algorithm. This algorithm is replaced by the interoperable specification in Section 4 of this document, which MUST be used by Server Cookie implementations.

This document has suggestions on Client Cookie construction in Section 3. The previous example in Appendix A.2 of [RFC7873] is NOT RECOMMENDED.

3. Constructing a Client Cookie

The Client Cookie acts as an identifier for a given client and its IP address, and needs to be unguessable. In order to provide minimal authentication of the targeted server, a client MUST use a different Client Cookie for each different Server IP address. This complicates a server's ability to spoof answers for other DNS servers. The Client Cookie SHOULD have 64-bits of entropy.

When a server does not support DNS Cookies, the client MUST NOT send the same Client Cookie to that same server again. Instead, it is recommended that the client does not send a Client Cookie to that server for a certain period, for example five minutes, before it retries with a new Client Cookie.

When a server does support DNS Cookies, the client should store the Client Cookie alongside the Server Cookie it registered for that server.

Except for when the Client IP address changes, there is no need to change the Client Cookie often. It is reasonable to change the Client Cookie then only if it has been compromised or after a relatively long implementation-defined period of time. The time period should be no longer than a year, and in any case Client Cookies are not expected to survive a program restart.

Client-Cookie = 64 bits of entropy

Previously, the recommended algorithm to compute the Client Cookie included Client IP address as an input to a hashing function. However, when implementing the DNS Cookies, several DNS vendors found impractical to include the Client IP as the Client Cookie is typically computed before the Client IP address is known. Therefore, the requirement to put Client IP address as input was removed.

However, for privacy reasons, in order to prevent tracking of devices across links and to not circumvent IPv6 Privacy Extensions [RFC4941], clients **MUST NOT** re-use a Client or Server Cookie after the Client IP address has changed.

One way to satisfy this requirement for non-re-use is to register the Client IP address alongside the Server Cookie when it receives the Server Cookie. In subsequent queries to the server with that Server Cookie, the socket **MUST** be bound to the Client IP address that was also used (and registered) when it received the Server Cookie. Failure to bind **MUST** then result in a new Client Cookie.

4. Constructing a Server Cookie

The Server Cookie is effectively a Message Authentication Code (MAC). The Server Cookie, when it occurs in a COOKIE option in a request, is intended to weakly assure the server that the request came from a client that is both at the source IP address of the request and using the Client Cookie included in the option. This assurance is provided by the Server Cookie that the server (or any other server from the anycast set) sent to that client in an earlier response appearing as the Server Cookie field in the request (see Section 5.2 of [RFC7873]).

DNS Cookies do not provide protection against "on-path" adversaries (see Section 9 of [RFC7873]). An on path observer that has seen a Server Cookie for a client, can abuse that Server Cookie to spoof request for that client within the timespan a Server Cookie is valid (see Section 4.3).

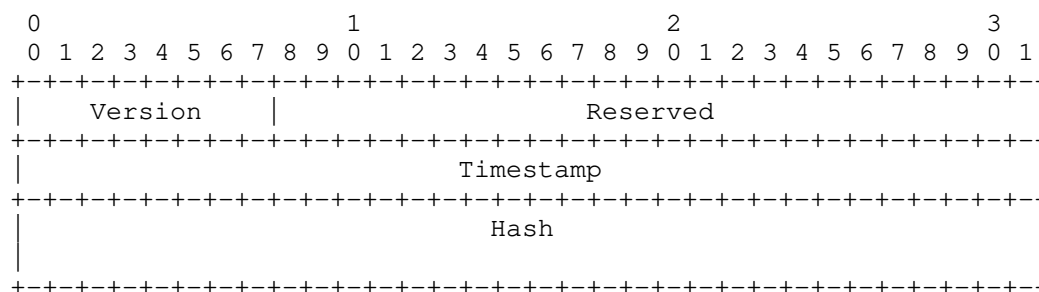
The Server Cookie is calculated from the Client Cookie, a series of Sub-Fields specified below, the Client IP address, and a Server Secret known only to the server, or servers responding on the same address in an anycast set.

For calculation of the Server Cookie, a pseudorandom function is **RECOMMENDED** with the property that an attacker that does not know the Server Secret, cannot find (any information about) the Server Secret and cannot create a Server Cookie for any combination of - the Client Cookie, the series of Sub-Fields specified below and the client IP address - for which it has not seen a Server Cookie before. Because

DNS servers need to calculate in order to verify Server Cookies, it is RECOMMENDED for the pseudorandom function to be performant. The [SipHash-2-4] pseudorandom function introduced in Section 4.4 fit these recommendations.

Changing the Server Secret regularly is RECOMMENDED but, when a secure pseudorandom function is used, it need not be changed too frequently. For example once a month would be adequate. See Section 5 on operator and implementation guidelines for updating a Server Secret.

The 128-bit Server Cookie consists of Sub-Fields: a 1 octet Version Sub-Field, a 3 octet Reserved Sub-Field, a 4 octet Timestamp Sub-Field and an 8 octet Hash Sub-Field.



4.1. The Version Sub-Field

The Version Sub-Field prescribes the structure and Hash calculation formula. This document defines Version 1 to be the structure and way to calculate the Hash Sub-Field as defined in this Section.

4.2. The Reserved Sub-Field

The value of the Reserved Sub-Field is reserved for future versions of server side Cookie construction. On construction it MUST be set to zero octets. On Server Cookie verification the server MUST NOT enforce those fields to be zero and the Hash should be computed with the received value as described in Section 4.4.

4.3. The Timestamp Sub-Field

The Timestamp value prevents Replay Attacks and MUST be checked by the server to be within a defined period of time. The DNS server SHOULD allow Cookies within 1 hour period in the past and 5 minutes into the future to allow operation of low volume clients and some limited time skew between the DNS servers in the anycast set.

The Timestamp value specifies a date and time in the form of a 32-bit *unsigned* number of seconds elapsed since 1 January 1970 00:00:00 UTC, ignoring leap seconds, in network byte order. All comparisons involving these fields MUST use "Serial number arithmetic", as defined in [RFC1982]. The [RFC1982] specifies how the differences should be handled. This handles any relative time window less than 68 years, at any time in the future (2038 or 2106 or 2256 or 22209 or later.)

The DNS server SHOULD generate a new Server Cookie at least if the received Server Cookie from the client is more than half an hour old, but MAY generate a new cookie more often than that.

4.4. The Hash Sub-Field

It's important that all the DNS servers use the same algorithm for computing the Server Cookie. This document defines the Version 1 of the server side algorithm to be:

```
Hash = SipHash-2-4(  
    Client Cookie | Version | Reserved | Timestamp | Client-IP,  
    Server Secret )
```

where "|" indicates concatenation.

Notice that Client-IP is used for hash generation even though it is not included in the cookie value itself. Client-IP can be either 4 bytes for IPv4 or 16 bytes for IPv6. The length of all the concatenated elements (the input into [SipHash-2-4]) MUST be either precisely 20 bytes in case of an IPv4 Client-IP or precisely 32 bytes in case of an IPv6 Client-IP.

When a DNS server receives a Server Cookie version 1 for validation, the length of the received COOKIE option MUST be precisely 24 bytes: 8 bytes for the Client Cookie plus 16 bytes for the Server Cookie. Verification of the length of the received COOKIE option is REQUIRED to guarantee the length of the input into [SipHash-2-4] to be precisely 20 bytes in case of an IPv4 Client-IP and precisely 32 bytes in case of an IPv6 Client-IP. This ensures that the input into [SipHash-2-4] is an injective function of the elements making up the input, and thereby prevents data substitution attacks. More specifically, this prevents a 36 byte COOKIE option coming from an IPv4 Client-IP to be validated as if it were coming from an IPv6 Client-IP.

The Server Secret MUST be configurable to make sure that servers in an anycast network return consistent results.

5. Updating the Server Secret

Changing the Server Secret regularly is RECOMMENDED. All servers in an anycast set must be able to verify the Server Cookies constructed by all other servers in that anycast set at all times. Therefore it is vital that the Server Secret is shared among all servers before it is used to generate Server Cookies on any server.

Also, to maximize maintaining established relationships between clients and servers, an old Server Secret should be valid for verification purposes for a specific period.

To facilitate this, deployment of a new Server Secret MUST be done in three stages:

Stage 1

The new Server Secret is deployed on all the servers in an anycast set by the operator.

Each server learns the new Server Secret, but keeps using the previous Server Secret to generate Server Cookies.

Server Cookies constructed with the both the new Server Secret and with the previous Server Secret are considered valid when verifying.

After stage 1 completed, all the servers in the anycast set have learned the new Server Secret, and can verify Server Cookies constructed with it, but keep generating Server Cookies with the old Server Secret.

Stage 2

This stage is initiated by the operator after the Server Cookie is present on all members in the anycast set.

When entering Stage 2, servers start generating Server Cookies with the new Server Secret. The previous Server Secret is not yet removed/forgotten about.

Server Cookies constructed with the both the new Server Secret and with the previous Server Secret are considered valid when verifying.

Stage 3

This stage is initiated by the operator when it can be assumed that most clients have obtained a Server Cookie derived from the new Server Secret.

With this stage, the previous Server Secret can be removed and MUST NOT be used anymore for verifying.

We RECOMMEND the operator to wait at least a period to be the longest TTL in the zones served by the server plus 1 hour after it initiated Stage 2, before initiating Stage 3.

The operator SHOULD wait at least longer than the period clients are allowed to use the same Server Cookie, which SHOULD be 1 hour, see Section 4.3.

6. Cookie Algorithms

[SipHash-2-4] is a pseudorandom function suitable as Message Authentication Code. This document REQUIRES compliant DNS server to use SipHash-2-4 as a mandatory and default algorithm for DNS Cookies to ensure interoperability between the DNS Implementations.

The construction method and pseudorandom function used in calculating and verifying the Server Cookies are determined by the initial version byte and by the length of the Server Cookie. Additional pseudorandom or construction algorithms for Server Cookies might be added in the future.

7. IANA Considerations

IANA is requested to create a registry on the "Domain Name System (DNS) Parameters" IANA web page as follows:

Registry Name: DNS Server Cookie Methods

Assignment Policy: Expert Review

Reference: [this document], [RFC7873]

Note: Server Cookie method (construction and pseudorandom algorithm) are determined by the Version in the first byte of the Cookie and by the Cookie size. Server Cookie size is limited to the inclusive range of 8 to 32 bytes.

Version	Size	Method
0	8-32	reserved
1	8-15	unassigned
1	16	SipHash-2-4 [this document] Section 4
1	17-32	unassigned
2-239	8-32	unassigned
240-254	8-32	private use
255	8-32	reserved

Table 1

8. Security and Privacy Considerations

DNS Cookies provide limited protection to DNS servers and clients against a variety of denial of service amplification, forgery or cache poisoning attacks by off-path attackers. They provide no protection against on-path adversaries that can observe the plaintext DNS traffic. An on-path adversary that can observe a Server Cookie for a client and server interaction, can use that Server Cookie for denial of service amplification, forgery or cache poisoning attacks directed at that client for the lifetime of the Server Cookie.

8.1. Client Cookie construction

In [RFC7873] it was RECOMMENDED to construct a Client Cookie by using a pseudorandom function of the Client IP address, the Server IP address, and a secret quantity known only to the client. The Client IP address was included to ensure that a client could not be tracked if its IP address changes due to privacy mechanisms or otherwise.

In this document, we changed Client Cookie construction to be just 64 bits of entropy newly created for each new upstream server the client connects to. As a consequence additional care needs to be taken to prevent tracking of clients. To prevent tracking, a new Client Cookie for a server MUST be created whenever the Client IP address changes.

Unfortunately, tracking Client IP address changes is impractical with servers that do not support DNS Cookies. To prevent tracking of clients with non DNS Cookie supporting servers, a client MUST NOT send a previously sent Client Cookie to a server not known to support DNS Cookies. To prevent the creation of a new Client Cookie for each query to an non DNS Cookies supporting server, it is RECOMMENDED to not send a Client Cookie to that server for a certain period, for example five minutes.

Summarizing:

- * In order to provide minimal authentication, a client MUST use a different Client Cookie for each different Server IP address.
- * To prevent tracking of clients, a new Client Cookie MUST be created when the Client IP address changes.
- * To prevent tracking of clients by a non DNS Cookie supporting server, a client MUST NOT send a previously sent Client Cookie to a server in the absence of an associated Server Cookie.

Note that it is infeasible for a client to detect change of the public IP address when the client is behind a routing device performing Network Address Translation (NAT). A server may track the public IP address of that routing device performing the NAT. Preventing tracking of the public IP of a NAT performing routing device is beyond the scope of this document.

8.2. Server Cookie construction

[RFC7873] did not give a precise recipe for constructing Server Cookies, but did recommend usage of a pseudorandom function strong enough to prevent guessing of cookies. In this document SipHash-2-4 is assigned as the pseudorandom function to be used for version 1 Server Cookies. SipHash-2-4 is considered sufficiently strong for the immediate future, but predictions about future development in cryptography and cryptanalysis are beyond the scope of this document.

The precise structure of version 1 Server Cookies is defined in this document. Portion of the structure is made up of unhashed data elements which are exposed in clear text to an on-path observer. These unhashed data elements are taken along as input to the SipHash-2-4 function of which the result is the other portion of the Server Cookie, so the unhashed portion of the Server Cookie can not be changed by an on-path attacker without also recalculating the hashed portion for which the Server Secret needs to be known.

One of the elements in the unhashed portion of version 1 Server Cookies is a Timestamp used to prevent Replay Attacks. Servers verifying version 1 Server Cookies need to have access to a reliable time value to compare with the Timestamp value, that cannot be altered by an attacker. Furthermore, all servers participating in an anycast set that validate version 1 Server Cookies need to have their clocks synchronized.

The cleartext Timestamp data element reveal to an on-path adversary using an observed Server Cookie to attack the client for which the Server Cookie was constructed (as shown in the first paragraph of this Section), the lifetime the observed Server Cookie can be used for the attack.

In addition to the Security Considerations in this section, the Security Considerations section of [RFC7873] still apply.

9. Acknowledgements

Thanks to Witold Krecicki and Pieter Lexis for valuable input, suggestions and text and above all for implementing a prototype of an interoperable DNS Cookie in Bind9, Knot and PowerDNS during the hackathon of IETF104 in Prague. Thanks for valuable input and suggestions go to Ralph Dolmans, Bob Harold, Daniel Salzman, Martin Hoffmann, Mukund Sivaraman, Petr Spacek, Loganaden Velvindron, Bob Harold, Philip Homburg, Tim Wicinski and Brian Dickson.

10. Normative References

- [RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <<https://www.rfc-editor.org/info/rfc1982>>.
- [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/info/rfc2119>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC7873] Eastlake 3rd, D. and M. Andrews, "Domain Name System (DNS) Cookies", RFC 7873, DOI 10.17487/RFC7873, May 2016, <<https://www.rfc-editor.org/info/rfc7873>>.

[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/info/rfc8174>>.

[SipHash-2-4]
Aumasson, J. and D. J. Bernstein, "SipHash: a fast short-input PRF", Progress in Cryptology - INDOCRYPT 2012. Lecture Notes in Computer Science, vol 7668. Springer., 2012, <https://doi.org/10.1007/978-3-642-34931-7_28>.

11. Informative References

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.

Appendix A. Test vectors

A.1. Learning a new Server Cookie

A resolver (client) sending from IPv4 address 198.51.100.100, sends a query for "example.com" to an authoritative server listening on 192.0.2.53 from which it has not yet learned the server cookie.

The DNS requests and replies shown in this Appendix, are in a "dig" like format. The content of the DNS COOKIE Option is shown in hexadecimal format after "; COOKIE:".

```
;; Sending:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57406
;; flags:; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2464c4abcf10c957
;; QUESTION SECTION:
;example.com.                IN      A

;; QUERY SIZE: 52
```

The authoritative nameserver (server) is configured with the following secret: e5e973e5a6b2a43f48e7dc849e37bfcf (as hex data).

It receives the query at Wed Jun 5 10:53:05 UTC 2019.

The content of the DNS COOKIE Option that the server will return is shown below in hexadecimal format after "; COOKIE:".

The Timestamp field Section 4.3 in the returned Server Cookie has value 1559731985. In [RFC3339] format this is 2019-06-05 10:53:05+00:00.

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57406
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2464c4abcf10c957010000005cf79f111f8130c3eee29480 (good)
;; QUESTION SECTION:
;example.com.                IN      A

;; ANSWER SECTION:
example.com.                 86400   IN      A      192.0.2.34

;; Query time: 6 msec
;; SERVER: 192.0.2.53#53(192.0.2.53)
;; WHEN: Wed Jun  5 10:53:05 UTC 2019
;; MSD SIZE  rcvd: 84
```

A.2. The same client learning a renewed (fresh) Server Cookie

40 minutes later, the same resolver (client) queries the same server for "example.org". It reuses the Server Cookie it learned in the previous query.

The Timestamp field in that previously learned Server Cookie, which is now send along in the request, was and is 1559731985. In [RFC3339] format this is 2019-06-05 10:53:05+00:00.

```
;; Sending:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50939
;; flags:; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2464c4abcf10c957010000005cf79f111f8130c3eee29480
;; QUESTION SECTION:
;example.org.                IN      A

;; QUERY SIZE: 52
```

The authoritative nameserver (server) now generates a new Server Cookie. The server SHOULD do this because it can see the Server Cookie send by the client is older than half an hour Section 4.3, but it is also fine for a server to generate a new Server Cookie sooner, or even for every answer.

The Timestamp field in the returned new Server Cookie has value 1559734385, which in [RFC3339] format is 2019-06-05 11:33:05+00:00.

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50939
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2464c4abcf10c957010000005cf7a871d4a564a1442aca77 (good)
;; QUESTION SECTION:
;example.org.                IN      A

;; ANSWER SECTION:
example.org.                 86400   IN      A      192.0.2.34

;; Query time: 6 msec
;; SERVER: 192.0.2.53#53(192.0.2.53)
;; WHEN: Wed Jun  5 11:33:05 UTC 2019
;; MSD SIZE  rcvd: 84
```

A.3. Another client learning a renewed Server Cookie

Another resolver (client) with IPv4 address 203.0.113.203 sends a request to the same server with a valid Server Cookie that it learned before (at Wed Jun 5 09:46:25 UTC 2019).

The Timestamp field in Server Cookie in the request has value 1559727985, which in [RFC3339] format is 2019-06-05 09:46:25+00:00.

Note that the Server Cookie has Reserved bytes set, but is still valid with the configured secret; the Hash part is calculated taking along the Reserved bytes.

```
;; Sending:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34736
;; flags:; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: fc93fc62807ddb8601abcdef5cf78f71a314227b6679ebf5
;; QUESTION SECTION:
;example.com.                IN      A

;; QUERY SIZE: 52
```

The authoritative nameserver (server) replies with a freshly generated Server Cookie for this client conformant with this specification; so with the Reserved bits set to zero.

The Timestamp field in the returned new Server Cookie has value 1559734700, which in [RFC3339] format is 2019-06-05 11:38:20+00:00.

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34736
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: fc93fc62807ddb86010000005cf7a9acf73a7810aca2381e (good)
;; QUESTION SECTION:
;example.com.                IN      A

;; ANSWER SECTION:
example.com.                 86400   IN      A      192.0.2.34

;; Query time: 6 msec
;; SERVER: 192.0.2.53#53(192.0.2.53)
;; WHEN: Wed Jun  5 11:38:20 UTC 2019
;; MSD SIZE  rcvd: 84
```

A.4. IPv6 query with rolled over secret

The query below is from a client with IPv6 address 2001:db8:220:1:59de:d0f4:8769:82b8 to a server with IPv6 address 2001:db8:8f::53. The client has learned a valid Server Cookie before (at Wed Jun 5 13:36:57 UTC 2019) when the Server had the secret: dd3bdf9344b678b185a6f5cb60fca715. The server now uses a new secret, but it can still validate the Server Cookie provided by the client as the old secret has not expired yet.

The Timestamp field in the Server Cookie in the request has value 1559741817, which in [RFC3339] format is 2019-06-05 13:36:57+00:00.

```
;; Sending:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6774
;; flags: qr aa; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
; COOKIE: 22681ab97d52c298010000005cf7c57926556bd0934c72f8
;; QUESTION SECTION:
;example.net.                IN      A

;; QUERY SIZE: 52
```

The authoritative nameserver (server) replies with a freshly generated server cookie for this client with its new secret: 445536bcd2513298075a5d379663c962

The Timestamp field in the returned new Server Cookie has value 1559741961, which in [RFC3339] format is .

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6774
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
; COOKIE: 22681ab97d52c298010000005cf7c609a6bb79d16625507a (good)
;; QUESTION SECTION:
;example.net.                IN      A

;; ANSWER SECTION:
example.net.                 86400   IN      A      192.0.2.34

;; Query time: 6 msec
;; SERVER: 2001:db8:8f::53#53(2001:db8:8f::53)
;; WHEN: Wed Jun  5 13:36:57 UTC 2019
;; MSD SIZE  rcvd: 84
```

Appendix B. Implementation status

At the time of writing, BIND from version 9.16 and Knot DNS from version 2.9.0 create Server Cookies according to the recipe described in this draft. Unbound and NSD have an Proof of Concept implementation that has been tested for interoperability during the hackathon at the IETF104 in Prague. Construction of privacy maintaining Client Cookies according to the directions in this draft

have been implemented in the getdns library and will be in the upcoming getdns-1.6.1 release and in Stubby version 0.3.1.

Authors' Addresses

Ondrej Sury
Internet Systems Consortium
Czechia

Email: ondrej@isc.org

Willem Toorop
NLnet Labs
Science Park 400
1098 XH Amsterdam
Netherlands

Email: willem@nlnetlabs.nl

Donald E. Eastlake 3rd
Futurewei Technologies
1424 Pro Shop Court
Davenport, FL 33896
United States of America

Phone: +1-508-333-2270
Email: d3e3e3@gmail.com

Mark Andrews
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063
United States of America

Email: marka@isc.org