

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.

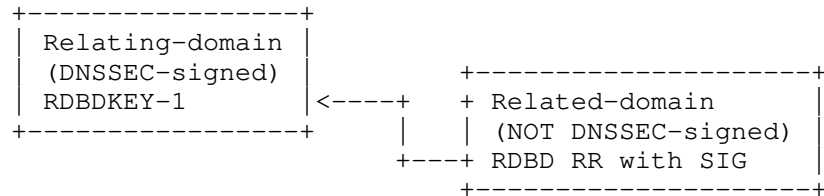
4.2. Extending DNSSEC

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-----
MIIEpAIBAACAQEAuZsJl5s8TmEPIx2vvYKV1bbZR166jfHP9JsIuZp2gV5mDCQ7
jOcXXMmFe+AIR8/4ZcqB5Wrw7BgTpDeHkC6LJWC2QBbEyOZCYre46uLm9zXhGGI3
//SRECJ7afvO+hz933/fBS8lCHG7A74RRJOo4pqV0EtQuembWOQOcDgThMFZ0C14
Hmg3eRwurQxUfn/7CqGYsq7yWcQic6aa9PIsdDmXLtBS1KWBiv4gMRWWNokES0y7
22z5D/GGZjBZOqliv3Lm9UC9k4AcV4H9l0SB+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
Gg3XVC1cb4vvrvDrkxY5aTDmizvVvF0MletBiLYjCwWHsOGql4hxwhvENYcYvCjs
T0WShG8FuuuHaH371+2hBkREeLHQRLyh0om2c8ECgYEA4JCb5PSNnRjB19hZWtzc
eGBu8lqVPNMqA1lMnQMe8qlJZsLj0mskIHd4N6Ez0eKyrJAcZjKfZwefzPaecOB3
/bNMQJhDSulcTXtFzjq0HdzAIR87FcnJ3iegTi1R0iKk/ymRuLGUodNa1u+85DB
7XYsy3f/LZoAESasJCWay6kCgYAYGpuc5BvwY5iF5FK/LMVZuH+OuHAF801hI8tg
GI5m/cS7EHD0+aVV38ivYdgRLpowIg4aOCxb19AI2j6KdAbeghsgpzyLx5sjmfYBt
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

DNS Operations
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2020

T. Finch
University of Cambridge
E. Hunt
ISC
P. van Dijk
PowerDNS
A. Eden
DNSimple
W. Mekking
ISC
July 8, 2019

Address-specific DNS aliases (ANAME)
draft-ietf-dnsop-aname-04

Abstract

This document defines the "ANAME" DNS RR type, to provide similar functionality to CNAME, but only for address queries. Unlike CNAME, an ANAME can coexist with other record types. The ANAME RR allows zone owners to make an apex domain name into an alias in a standards compliant manner.

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 January 9, 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
 - 1.1. Overview
 - 1.2. Terminology
2. The ANAME resource record
 - 2.1. Presentation and wire format

- 2.2. Coexistence with other types
- 3. Substituting ANAME sibling address records
- 4. ANAME processing by primary masters
 - 4.1. Zone transfers
 - 4.2. DNSSEC
 - 4.3. TTLs
- 5. ANAME processing by resolvers
- 6. Query processing
 - 6.1. Authoritative servers
 - 6.1.1. Address queries
 - 6.1.2. ANAME queries
 - 6.2. Resolvers
 - 6.2.1. Address queries
 - 6.2.2. ANAME queries
- 7. IANA considerations
- 8. Security considerations
- 9. Acknowledgments
- 10. Changes since the last revision
 - 10.1. Version -04
 - 10.2. Version -03
 - 10.3. Version -02
- 11. References
 - 11.1. Normative References
 - 11.2. Informative References
 - 11.3. URIs
- Appendix A. Implementation status
- Appendix B. Historical note
- Appendix C. On preserving TTLs
 - C.1. Query bunching
 - C.2. Upstream caches
 - C.3. ANAME chains
 - C.4. ANAME substitution inside the name server
 - C.5. TTLs and zone transfers
- Appendix D. Alternative setups
 - D.1. Reducing query volume
 - D.2. Zone transfer scalability
 - D.3. Tailored responses
- Appendix E. ANAME loops
- Authors' Addresses

1. Introduction

It can be desirable to provide web sites (and other services) at a bare domain name (such as "example.com") as well as a service-specific subdomain ("www.example.com").

If the web site is hosted by a third-party provider, the ideal way to provision its name in the DNS is using a CNAME record, so that the third party provider retains control over the mapping from names to IP address(es). It is now common for name-to-address mappings to be highly dynamic, dependent on client location, server load, etc.

However, CNAME records cannot coexist with other records with the same owner name. (The reason why is explored in Appendix B). This restriction means they cannot appear at a zone apex (such as "example.com") because of the SOA, NS, and other records that have to be present there. CNAME records can also conflict at subdomains, for example, if "department.example.edu" has separately hosted mail and web servers.

Redirecting website lookups to an alternate domain name via SRV or URI resource records would be an effective solution from the DNS point of view, but to date, browser vendors have not accepted this approach.

As a result, the only widely supported and standards-compliant way to publish a web site at a bare domain is to place address records (A

and/or AAAA) at the zone apex. The flexibility afforded by CNAME is not available.

This document specifies a new RR type "ANAME", which provides similar functionality to CNAME, but only for address queries (i.e., for type A or AAAA). The basic idea is that the address records next to an ANAME record are automatically copied from and kept in sync with the ANAME target's address records. The ANAME record can be present at any DNS node, and can coexist with most other RR types, enabling it to be present at a zone apex, or any other name where the presence of other records prevents the use of a CNAME record.

Similar authoritative functionality has been implemented and deployed by a number of DNS software vendors and service providers, using names such as ALIAS, ANAME, apex CNAME, CNAME flattening, and top-level redirection. These mechanisms are proprietary, which hinders the ability of zone owners to have the same data served from multiple providers or to move from one provider to another. None of these proprietary implementations includes a mechanism for resolvers to follow the redirection chain themselves.

1.1. Overview

The core functionality of this mechanism allows zone administrators to start using ANAME records unilaterally, without requiring secondary servers or resolvers to be upgraded.

- o The resource record definition in Section 2 is intended to provide zone data portability between standards-compliant DNS servers and the common core functionality of existing proprietary ANAME-like facilities.
- o The zone maintenance mechanism described in Section 4 keeps the ANAME's sibling address records in sync with the ANAME target.

This definition is enough to be useful by itself. However, it can be less than optimal in certain situations: for instance, when the ANAME target uses clever tricks to provide different answers to different clients to improve latency or load balancing. The query processing rules in Section 6 require to include the ANAME record so that resolvers can use this information (as described in Section 5) to obtain answers that are tailored to the resolver rather than to the zone's primary master.

Resolver support for ANAME is not necessary, since ANAME-oblivious resolvers can get working answers from authoritative servers. It's just an optimization that can be rolled out incrementally, and that will help ANAME to work better the more widely it is deployed.

1.2. Terminology

An "address record" is a DNS resource record whose type is A or AAAA. These are referred to as "address types". "Address query" refers to a DNS query for any address type.

When talking about "address records" we mean the entire RRset, including owner name and TTL. We treat missing address records (i.e. NXDOMAIN or NODATA) the same successfully resolving as a set of zero address records, and distinct from "failure" which covers error responses such as SERVFAIL or REFUSED.

The "sibling address records" of an ANAME record are the address records at the same owner name as the ANAME, which are subject to ANAME substitution.

The "target address records" of an ANAME record are the address records obtained by resolving the ultimate target of the ANAME (see

Section 3).

During the process of looking up the target address records, one or more CNAME or ANAME records may be encountered. These records are not the final target address records, and are referred in this document as "intermediate records". The target name must be replaced with the new name provided in the RDATA and the new target is resolved.

Other DNS-related terminology can be found in [RFC8499].

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in [RFC2119].

2. The ANAME resource record

This document defines the "ANAME" DNS resource record type, with RR TYPE value [TBD].

2.1. Presentation and wire format

The ANAME presentation format is identical to that of CNAME [RFC1033]:

```
owner ttl class ANAME target
```

The wire format is also identical to CNAME [RFC1035], except that name compression is not permitted in ANAME RDATA, per [RFC3597].

2.2. Coexistence with other types

Only one ANAME <target> can be defined per <owner>. An ANAME RRset MUST NOT contain more than one resource record.

An ANAME's sibling address records are under the control of ANAME processing (see Section 4) and are not first-class records in their own right. They MAY exist in zone files, but they can subsequently be altered by ANAME processing.

An ANAME record MAY freely coexist at the same owner name with other RR types, except they MUST NOT coexist with CNAME or any other RR type that restricts the types with which it can itself coexist. That means An ANAME record can coexist at the same owner name with A and AAAA records. These are the sibling address records that are updated with the target addresses that are retrieved through the ANAME substitution process Section 3.

Like other types, An ANAME record can coexist with DNAME records at the same owner name; in fact, the two can be used cooperatively to redirect both the owner name address records (via ANAME) and everything under it (via DNAME).

3. Substituting ANAME sibling address records

This process is used by both primary masters (see Section 4) and resolvers (see Section 5), though they vary in how they apply the edit described in the final step. However, this process is not exclusively used by primary masters and resolvers: it may be executed as a bump in the wire, as part of the query lookup, or at any other point during query resolution.

The following steps MUST be performed for each address type:

1. Starting at the ANAME owner, follow the chain of ANAME and/or CNAME records as far as possible to find the ultimate target.

2. If a loop is detected, continue with an empty RRset, otherwise get the ultimate target's address records. (Ignore any sibling address records of intermediate ANAMES.)
3. Stop if resolution failed. (Note that NXDOMAIN and NODATA count as successfully resolving an empty RRset.)
4. If one or more address records are found, replace the owner of the target address records with the owner of the ANAME record. Set the TTL to the minimum of the ANAME TTL, the TTL of each intermediate record, and the TTL of the target address records. Drop any RRSIG records.
5. Stop if this modified RRset is the same as the sibling RRset (ignoring any RRSIG records). The comparison MAY treat nearly-equal TTLs as the same.
6. Delete the sibling address RRset (if any) and replace it with the modified RRset.

At this point, the substituted RRset is not signed. A primary master will proceed to sign the substituted RRset, whereas resolvers can only use the substituted RRset when an unsigned answer is appropriate. This is explained in more detail in the following sections.

4. ANAME processing by primary masters

Each ANAME's sibling address records are kept up-to-date as if by the following process, for each address type:

- o Perform ANAME sibling address record substitution as described in Section 3. Any edit performed in the final step is applied to the ANAME's zone. A primary server MAY use Dynamic Updates (DNS UPDATE) [RFC2136] to update the zone.
- o If resolution failed, wait for a period before trying again. This retry time SHOULD be configurable.
- o Otherwise, wait until the target address RRset TTL has expired or is close to expiring, then repeat.

It may be more efficient to manage the polling per ANAME target rather than per ANAME as specified (for example if the same ANAME target is used by multiple zones).

Sibling address records are committed to the zone and stored in nonvolatile storage. This allows a server to restart without delays due to ANAME processing, use offline DNSSEC signing, and not implement special ANAME processing logic when handling a DNS query.

Appendix D describes how ANAME would fit in different DNS architectures that use online signing or tailored responses.

4.1. Zone transfers

ANAME is no more special than any other RRtype and does not introduce any special processing related to zone transfers.

A zone containing ANAME records that point to frequently-changing targets will itself change frequently, and may see an increased number of zone transfers. Or if a very large number of zones are sharing the same ANAME target, and that changes address, that may cause a great volume of zone transfers. Guidance on dealing with ANAME in large scale implementations is provided Appendix D.

Secondary servers rely on zone transfers to obtain sibling address

records, just like the rest of the zone, and serve them in the usual way (see Section 6). A working DNS NOTIFY [RFC1996] setup is recommended to avoid extra delays propagating updated sibling address records when they change.

4.2. DNSSEC

A zone containing ANAME records that will update address records has to do so before signing the zone with DNSSEC [RFC4033] [RFC4034] [RFC4035]. This means that for traditional DNSSEC signing the substitution of sibling address records must be done before signing and loading the zone into the name server. For servers that support online signing, the substitution may happen as part of the name server process, after loading the zone.

DNSSEC signatures on sibling address records are generated in the same way as for normal (dynamic) updates.

4.3. TTLs

Sibling address records are served from authoritative servers with a fixed TTL. Normally this TTL is expected to be the same as the target address records' TTL; however the exact mechanism for obtaining the target is unspecified, so cache effects, following ANAME and CNAME chains, or deliberate policies might make the sibling TTL smaller.

This means that when adding address records into the zone as a result of ANAME processing, the TTL to use is at most that of the TTL of the address target records. If you use a higher value, this will stretch the TTL which is undesired.

TTL stretching is hard to avoid when implementing ANAME substitution at the primary: The target address records' TTL influences the update rate of the zone, while the sibling address records' TTL determine how long a resolver may cache the address records. Thus, the end-to-end TTL (from the authoritative servers for the target address records to end-user DNS caches) is nearing twice the target address record TTL. There is a more extended discussion of TTL handling in Appendix C.

5. ANAME processing by resolvers

When a resolver makes an address query in the usual way, it might receive a response containing ANAME information in the Answer section, as described in Section 6. This informs the resolver that it MAY resolve the ANAME target address records to get answers that are tailored to the resolver rather than the ANAME's primary master.

In order to provide tailored answers to clients that are ANAME-oblivious, the resolver MAY perform sibling address record substitution in the following situations:

- o The resolver's client queries with DO=0. (As discussed in Section 8, if the resolver finds it would downgrade a secure answer to insecure, it MAY choose not to substitute the sibling address records.)
- o The resolver's client queries with DO=1 and the ANAME and sibling address records are unsigned. (Note that this situation does not apply when the records are signed but insecure: the resolver might not be able to validate them because of a broken chain of trust, but its client could have an extra trust anchor that does allow it to validate them; if the resolver substitutes the sibling address records they will become bogus.)

In these first two cases, the resolver MAY perform ANAME sibling

address record substitution as described in Section 3. Any edit performed in the final step is applied to the Answer section of the response.

If the resolver's client is querying using an API such as "getaddrinfo" [RFC3493] that does not support DNSSEC validation, the resolver MAY perform ANAME sibling address record substitution as described in Section 3. Any edits performed in the final step are applied to the addresses returned by the API. (This case is for validating stub resolvers that query an upstream recursive server with DO=1, so they cannot rely on the recursive server to do ANAME substitution for them.)

6. Query processing

6.1. Authoritative servers

6.1.1. Address queries

When a server receives an address query for a name that has an ANAME record, the response's Answer section MUST contain the ANAME record, in addition to the sibling address queries. The ANAME record indicates to a client that it might wish to resolve the target address records itself.

6.1.2. ANAME queries

When a server receives an query for type ANAME, regardless of whether the ANAME record exists on the queried domain, any sibling address records SHOULD be added to the Additional section. Note that the sibling address records may have been substituted already.

When adding address records to the Additional section, if not all address types are present and the zone is signed, the server SHOULD include a DNSSEC proof of nonexistence for the missing address types.

6.2. Resolvers

6.2.1. Address queries

When a server receives an address query for a name that has an ANAME record, the response's Answer section MUST contain the ANAME record, in addition to the sibling address queries.

The Additional section MAY contain the target address records that match the query type (or the corresponding proof of nonexistence), if they are available in the cache and the target address RDATA fields differ from the sibling address RRset.

An ANAME target MAY resolve to address records via a chain of CNAME and/or ANAME records; any CNAME/ANAME chain MUST be included when adding target address records to a response's Additional section.

6.2.2. ANAME queries

When a resolver receives an query for type ANAME, any sibling address records SHOULD be added to the Additional section. Just like with an authoritative server, when adding address records to the Additional section, if not all address types are present and the zone is signed, the resolver SHOULD include a DNSSEC proof of nonexistence for the missing address types.

7. IANA considerations

IANA is requested to assign a DNS RR TYPE value for ANAME resource records under the "Resource Record (RR) TYPES" subregistry under the "Domain Name System (DNS) Parameters" registry.

IANA might wish to consider the creation of a registry of address types; addition of new types to such a registry would then implicitly update this specification.

8. Security considerations

When a primary master updates an ANAME's sibling address records to match its target address records, it uses its own best information as to the correct answer. The primary master might sign the updated records, but that is not a guarantee of the actual correctness of the answer. This signing can have the effect of promoting an insecure response from the ANAME <target> to a signed response from the <owner>, which can then appear to clients to be more trustworthy than it should. DNSSEC validation SHOULD be used when resolving the ANAME <target> to mitigate this possible harm. Primary masters MAY refuse to substitute ANAME sibling address records unless the <target> node is both signed and validated.

When a resolver substitutes an ANAME's sibling address records, it can find that the sibling address records are secure but the target address records are insecure. Going ahead with the substitution will downgrade a secure answer to an insecure one. However this is likely to be the counterpart of the situation described in the previous paragraph, so the resolver is downgrading an answer that the ANAME's primary master upgraded. A resolver will only downgrade an answer in this way when its client is security-oblivious; however the client's path to the resolver is likely to be practically safer than the resolver's path to the ANAME target's servers. Resolvers MAY choose not to substitute sibling address records when they are more secure than the target address records.

9. Acknowledgments

Thanks to Mark Andrews, Ray Bellis, Stefan Buehler, Paul Ebersman, Richard Gibson, Tatuya JINMEI, Hakan Lindqvist, Mattijs Mekking, Stephen Morris, Bjorn Mott, Richard Salts, Mukund Sivaraman, Job Snijders, Jan Vcelak, Paul Vixie, Duane Wessels, and Paul Wouters, Olli Vanhoja, Brian Dickson for discussion and feedback.

10. Changes since the last revision

[This section is to be removed before publication as an RFC.]

The full history of this draft and its issue tracker can be found at <https://github.com/each/draft-aname> [1]

10.1. Version -04

- o Split up section about Additional Section processing.
- o Update Additional Section processing requirements.
- o Clarify when ANAME resolution may happen [#43].
- o Revisit TTL considerations [#30, #34].
- o ANAME goes into the Answer section when QTYPE=A|AAAA [#62].
- o Update alternative setups section with concerns (Brian Dickson) [#68].
- o Add section on ANAME loops (open issue [#45]).

10.2. Version -03

- o Grammar improvements (Olli Vanhoja)

- o Split up Implications section, clarify text on zone transfers and dynamic updates [#39].
- o Rewrite Alternative setup section and move to Appendix, add text on zone transfer scalability concerns and GeoIP.

10.3. Version -02

Major revamp, so authoritative servers (other than primary masters) now do not do any special ANAME processing, just Additional section processing.

11. References

11.1. Normative References

- [RFC1033] Lottor, M., "Domain Administrators Operations Guide", RFC 1033, DOI 10.17487/RFC1033, November 1987, <<https://www.rfc-editor.org/info/rfc1033>>.
- [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>>.
- [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>>.
- [RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <<https://www.rfc-editor.org/info/rfc3597>>.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [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>>.
- [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>>.
- [RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <<https://www.rfc-editor.org/info/rfc7871>>.
- [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>>.

11.2. Informative References

- [RFC0882] Mockapetris, P., "Domain names: Concepts and facilities", RFC 882, DOI 10.17487/RFC0882, November 1983,

<<https://www.rfc-editor.org/info/rfc882>>.

- [RFC0973] Mockapetris, P., "Domain system changes and observations", RFC 973, DOI 10.17487/RFC0973, January 1986, <<https://www.rfc-editor.org/info/rfc973>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <<https://www.rfc-editor.org/info/rfc2308>>.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <<https://www.rfc-editor.org/info/rfc3493>>.

11.3. URIs

[1] <https://github.com/each/draft-aname>

[2] <https://github.com/each/draft-aname/issues/45>

Appendix A. Implementation status

PowerDNS currently implements a similar authoritative-only feature using "ALIAS" records, which are expanded by the primary server and transferred as address records to secondaries.

[TODO: Add discussion of DNSimple, DNS Made Easy, EasyDNS, Cloudflare, Amazon, Dyn, and Akamai.]

Appendix B. Historical note

In the early DNS [RFC0882], CNAME records were allowed to coexist with other records. However this led to coherency problems: if a resolver had no cache entries for a given name, it would resolve queries for un-cached records at that name in the usual way; once it had cached a CNAME record for a name, it would resolve queries for un-cached records using CNAME target instead.

For example, given the zone contents below, the original CNAME behaviour meant that if you asked for "alias.example.com TXT" first, you would get the answer "owner", but if you asked for "alias.example.com A" then "alias.example.com TXT" you would get the answer "target".

alias.example.com.	TXT	"owner"
alias.example.com.	CNAME	canonical.example.com.
canonical.example.com.	TXT	"target"
canonical.example.com.	A	192.0.2.1

This coherency problem was fixed in [RFC0973] which introduced the inconvenient rule that a CNAME acts as an alias for all other RR types at a name, which prevents the coexistence of CNAME with other records.

A better fix might have been to improve the cache's awareness of which records do and do not coexist with a CNAME record. However that would have required a negative cache mechanism which was not added to the DNS until later [RFC1034] [RFC2308].

While [RFC2065] relaxed the restriction by allowing coexistence of CNAME with DNSSEC records, this exception is still not applicable to other resource records. RRSIG and NSEC exist to prove the integrity of the CNAME record; they are not intended to associate arbitrary data with the domain name. DNSSEC records avoid interoperability problems by being largely invisible to security-oblivious resolvers.

Now that the DNS has negative caching, it is tempting to amend the algorithm for resolving with CNAME records to allow them to coexist with other types. Although an amended resolver will be compatible with the rest of the DNS, it will not be of much practical use because authoritative servers which rely on coexisting CNAMEs will not interoperate well with older resolvers. Practical experiments show that the problems are particularly acute when CNAME and MX try to coexist.

Appendix C. On preserving TTLs

An ANAME's sibling address records are in an unusual situation: they are authoritative data in the owner's zone, so from that point of view the owner has the last say over what their TTL should be; on the other hand, ANAMES are supposed to act as aliases, in which case the target should control the address record TTLs.

However there are some technical constraints that make it difficult to preserve the target address record TTLs.

The following subsections conclude that the end-to-end TTL (from the authoritative servers for the target address records to end-user DNS caches) is nearing twice the target address record TTL.

C.1. Query bunching

If the times of end-user queries for a domain name are well distributed, then (typically) queries received by the authoritative servers for that domain are also well distributed. If the domain is popular, a recursive server will re-query for it once every TTL seconds, but the periodic queries from all the various recursive servers will not be aligned, so the queries remain well distributed.

However, imagine that the TTLs of an ANAME's sibling address records are decremented in the same way as cache entries in recursive servers. Then all the recursive servers querying for the name would try to refresh their caches at the same time when the TTL reaches zero. They would become synchronized, and all the queries for the domain would be bunched into periodic spikes.

This specification says that ANAME sibling address records have a normal fixed TTL derived from (e.g. equal or nearly equal to) the target address records' original TTL. There is no cache-like decrementing TTL, so there is no bunching of queries.

C.2. Upstream caches

There are two straightforward ways to get an RRset's original TTL:

- o by directly querying an authoritative server;
- o using the original TTL field from the RRset's RRGIG record(s).

However, not all zones are signed, and a primary master might not be able to query other authoritative servers directly (e.g. if it is a

hidden primary behind a strict firewall). Instead it might have to obtain an ANAME's target address records via some other recursive server.

Querying via a separate recursive server means the primary master cannot trivially obtain the target address records' original TTLs. Fortunately this is likely to be a self-correcting problem for similar reasons to the query-bunching discussed in the previous subsection. The primary master can inspect the target address records just after the TTL expires when its upstream cache has just refreshed them, so the TTL will be nearly equal to the original TTL.

A related consideration is that the primary master cannot in general refresh its copies of an ANAME's target address records more frequently than their TTL, without privileged control over its resolver cache.

Combined with the requirement that sibling address records are served with a fixed TTL, this means that the end-to-end TTL will be the target address record TTL (which determines when the sibling address records are updated) plus the sibling address record TTL (which determines when end-user caches are updated). Since the sibling address record TTL is derived from the target address records' original TTL, the end-to-end TTL will be nearing twice the target address record TTL.

C.3. ANAME chains

ANAME sibling address record substitution is made slightly more complicated by the requirement to follow chains of ANAME and/or CNAME records. The TTL of the substituted address records is the minimum of TTLs of the ANAME, all the intermediate records, and target records. This stops the end-to-end TTL from being inflated by each ANAME in the chain.

With CNAME records, repeat queries for "cname.example. CNAME target.example." must not be fully answered from cache after its TTL expires, but must instead be sent to name servers authoritative for "cname.example" in case the CNAME has been updated or removed. Similarly, an ANAME at "aname.example" means that repeat queries for "aname.example" must not be fully answered from cache after its TTL expire, but must instead be sent to name servers authoritative for aname.example in case the ANAME has been updated or removed.

C.4. ANAME substitution inside the name server

When ANAME substitution is performed inside the authoritative name server (as described in #alternatives) or in the resolver (as described in #resolver) the end-to-end TTL will actually be just the target address record TTL.

An authoritative server that has control over its resolver can use a cached target address RRset and decremented TTL in the response to the client rather than using the original target address records' TTL. It SHOULD however not use TTLs in the response that are nearing zero to avoid query bunching Appendix C.1.

A resolver that performs ANAME substitution is able to get the original TTL from the authoritative name server and use its own cache to store the substituted address records with the appropriate TTL, thereby honoring the TTL of target address records.

C.5. TTLs and zone transfers

When things are working properly (with secondary name servers responding to NOTIFY messages promptly) the authoritative servers will follow changes to ANAME target address records according to

their TTLs. As a result the end-to-end TTL is unchanged from the previous subsection.

If NOTIFY doesn't work, the TTLs can be stretched by the zone's SOA refresh timer. More serious breakage can stretch them up to the zone expiry time.

Appendix D. Alternative setups

If you are a large scale DNS provider, ANAME may introduce some operational concerns.

D.1. Reducing query volume

When doing ANAME target lookups, an authoritative server might want to use longer TTLs to reduce query volume, for ANAME values that do not change frequently. This is the same concern a recursive resolver may be exposed to when receiving answers with short TTLs. An authoritative server doing ANAME target lookups therefor could use the same mitigation as a recursive nameserver, that is set a configured minimum TTL usage. This may however contribute to TTL stretching as described in Section 4.3 so the configured minimum should not be too low.

D.2. Zone transfer scalability

A frequently changing ANAME target, or a ANAME target that changes its address and is used for many zones, can lead to an increased number of zone transfers. Such DNS architectures may want to consider a zone transfer mechanism outside the DNS.

Another way to deal with zone transfer scalability is to move the ANAME processing (Section 3) inside the name server daemon. This is not a requirement for ANAME to work, but may be a better solution in large scale implementations. These implementations usually already rely on online DNSSEC signing for similar reasons. If ANAME processing occurs inside the name server daemon, it MUST be done before any DNSSEC online signing happens.

For example, some existing ANAME-like implementations are based on a DNS server architecture, in which a zone's published authoritative servers all perform the duties of a primary master in a distributed manner: provisioning records from a non-DNS back-end store, refreshing DNSSEC signatures, and so forth. They don't use standard zone transfers, and already implement their ANAME-like processing inside the name server daemon, substituting ANAME sibling address records on demand.

D.3. Tailored responses

Some DNS providers will tailor responses based on information in the client request. Such implementations will use the source IP address or EDNS Client Subnet [RFC7871] information and use geographical data (GeoIP) or network latency measurements to decide what the best answer is for a given query. Such setups won't work with traditional DNSSEC and provide DNSSEC support usually through online signing. Similar such setups should provide ANAME support through substituting ANAME sibling records on demand.

Also, an authoritative server that uses the client address to tailor the response should obviously not use its own address when looking up ANAME targets, or it could direct clients to a suboptimal server (e.g. a wrong language, or regional restricted content). Instead the authoritative server should look up the ANAME targets on behalf of the client address. It could use for example EDNS Client Subnet for this.

In short, the exact mechanism for obtaining the target address records in such setups is unspecified; typically they will be resolved in the DNS in the usual way, but if an ANAME implementation has special knowledge of the target it can short-cut the substitution process, or it can use clever tricks such as client-dependant answers to make the answer more optimal.

Appendix E. ANAME loops

The ANAME sibling address substitution algorithm in Section 3 poses a challenge of detecting a loop between two or more ANAME records. Imagine this setup: two authoritative servers X and Y performing ANAME sibling address substitution on the fly (i.e. they attempt to resolve the ANAME target when the client query arrives). If server X gets a query for FOO.TEST which is an ANAME to BAR.TEST, it will send a query to server Y for BAR.TEST which is an ANAME to FOO.TEST. Server Y will then start a new query to server X, which has no way to know that it is regarding the original FOO.TEST lookup.

The only indicator of the presence of the loop in the described setup is the network timeout. Ideally we would recognize the loop explicitly based on the exchanged DNS messages.

On-the-fly ANAME substitution is allowed and it's just the most obvious scenario where the problem can be demonstrated, but this loop can also be encountered in other situations. The root cause is that when the server gets a query it doesn't know why and that the server always attempts to fully resolve the ANAME target before sending the response.

TODO: Solve this issue [<https://github.com/each/draft-aname/issues/45> [2]]

Authors' Addresses

Tony Finch
University of Cambridge
University Information Services
Roger Needham Building
7 JJ Thomson Avenue
Cambridge CB3 0RB
England

Email: dot@dotat.at

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

Email: each@isc.org

Peter van Dijk
PowerDNS.COM B.V.
Den Haag
The Netherlands

Email: peter.van.dijk@powerdns.com

Anthony Eden
DNSimple
Boston, MA USA

Email: anthony.eden@dnsimple.com
URI: <https://dnsimple.com/>

Matthijs Mekking
ISC
950 Charter St
Redwood City, CA 94063
USA

Email: matthijs@isc.org

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, Tatuya 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: Informational
Expires: October 21, 2020

S. Huque
P. Aras
Salesforce
J. Dickinson
Sinodun
J. Vcelak
NSI
D. Blacka
Verisign
April 19, 2020

Multi Signer DNSSEC models
draft-ietf-dnsop-multi-provider-dnssec-05

Abstract

Many enterprises today employ the service of multiple DNS providers to distribute their authoritative DNS service. Deploying DNSSEC in such an environment may present some challenges depending on the configuration and feature set in use. In particular, when each DNS provider independently signs zone data with their own keys, additional key management mechanisms are necessary. This document presents deployment models that accommodate this scenario and describe these key management requirements. These models do not require any changes to the behavior of validating resolvers, nor do they impose the new key management requirements on authoritative servers not involved in multi signer configurations.

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 October 21, 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 Motivation	2
2. Deployment Models	3
2.1. Multiple Signer models	3
2.1.1. Model 1: Common KSK set, Unique ZSK set per provider	4
2.1.2. Model 2: Unique KSK set and ZSK set per provider . .	5
3. Validating Resolver Behavior	5
4. Signing Algorithm Considerations	6
5. Authenticated Denial Considerations	6
5.1. Single Method	7
5.2. Mixing Methods	7
6. Key Rollover Considerations	8
6.1. Model 1: Common KSK, Unique ZSK per provider	8
6.2. Model 2: Unique KSK and ZSK per provider	9
7. Using Combined Signing Keys	10
8. Use of CDS and CDNSKEY	10
9. Key Management Mechanism Requirements	10
10. DNS Response Size Considerations	11
11. IANA Considerations	11
12. Security Considerations	11
13. Acknowledgments	12
14. References	12
14.1. Normative References	12
14.2. Informative References	13
Authors' Addresses	14

1. Introduction and Motivation

RFC EDITOR: PLEASE REMOVE THIS PARAGRAPH BEFORE PUBLISHING: The source for this draft is maintained in GitHub at: <https://github.com/shuque/multi-provider-dnssec>

Many enterprises today employ the service of multiple Domain Name System (DNS) [RFC1034] [RFC1035] providers to distribute their authoritative DNS service. This is primarily done for redundancy and availability, and allows the DNS service to survive a complete, catastrophic failure of any single provider. Additionally, enterprises or providers occasionally have requirements that preclude standard zone transfer techniques [RFC1995] [RFC5936] : either non-standardized DNS features are in use that are incompatible with zone transfer, or operationally a provider must be able to (re)sign DNS records using their own keys. This document outlines some possible models of DNSSEC [RFC4033] [RFC4034] [RFC4035] deployment in such an environment.

This document assumes a reasonable level of familiarity with DNS operations and protocol terms. Much of the terminology is explained in further detail in DNS Terminology [RFC8499].

2. Deployment Models

If a zone owner can use standard zone transfer techniques, then the presence of multiple providers does not require modifications to the normal deployment models. In these deployments, there is a single signing entity (which may be the zone owner, one of the providers, or a separate entity), while the providers act as secondary authoritative servers for the zone.

Occasionally, however, standard zone transfer techniques cannot be used. This could be due to the use of non-standard DNS features, or due to operational requirements of a given provider (e.g., a provider that only supports "online signing".) In these scenarios, the multiple providers each act like primary servers, independently signing data received from the zone owner and serving it to DNS queriers. This configuration presents some novel challenges and requirements.

2.1. Multiple Signer models

In this category of models, multiple providers each independently sign and serve the same zone. The zone owner typically uses provider-specific APIs to update zone content identically at each of the providers, and relies on the provider to perform signing of the data. A key requirement here is to manage the contents of the DNSKEY and Delegation Signer (DS) RRsets in such a way that validating resolvers always have a viable path to authenticate the DNSSEC signature chain, no matter which provider is queried. This requirement is achieved by having each provider import the public Zone Signing Keys (ZSKs) of all other providers into their DNSKEY RRsets.

These models can support DNSSEC even for the non-standard features mentioned previously, if the DNS providers have the capability of signing the response data generated by those features. Since these responses are often generated dynamically at query time, one method is for the provider to perform online signing (also known as on-the-fly signing). However, another possible approach is to pre-compute all the possible response sets and associated signatures, and then algorithmically determine at query time which response set and signature needs to be returned.

In the models presented, the function of coordinating the DNSKEY or DS RRset does not involve the providers communicating directly with each other. Feedback from several commercial managed DNS providers indicates that they may be unlikely to directly communicate, since they typically have a contractual relationship only with the zone owner. However, if the parties involved are agreeable, it may be possible to devise a protocol mechanism by which the providers directly communicate to share keys. Details of such a protocol are deferred to a future specification document, should there be interest.

In the descriptions below, the Key Signing Key (KSK), and Zone Signing Key (ZSK), correspond to the definitions in [RFC8499], with the caveat that the KSK not only signs the zone apex DNSKEY RRset, but also serves as the Secure Entry Point (SEP) into the zone.

2.1.1. Model 1: Common KSK set, Unique ZSK set per provider

- o The zone owner holds the KSK set, manages the DS record set, and is responsible for signing the DNSKEY RRset and distributing it to the providers.
- o Each provider has their own ZSK set which is used to sign data in the zone.
- o The providers have an API that the zone owner uses to query the ZSK public keys, and insert a combined DNSKEY RRset that includes the ZSK sets of each provider, and the KSK set, signed by the KSK.
- o Note that even if the contents of the DNSKEY RRset do not change, the zone owner needs to periodically re-sign it as signature expiration approaches. The provider API is also used to thus periodically redistribute the refreshed DNSKEY RRset.
- o Key rollovers need coordinated participation of the zone owner to update the DNSKEY RRset (for KSK or ZSK), and the DS RRset (for KSK).

- o (One specific variant of this model that may be interesting is a configuration in which there is only a single provider. A possible use case for this is where the zone owner wants to outsource the signing and operation of their DNS zone to a single 3rd party provider, but still control the KSK, so that they can authorize and/or revoke the use of specific zone signing keys.)

2.1.2. Model 2: Unique KSK set and ZSK set per provider

- o Each provider has their own KSK and ZSK sets.
- o Each provider offers an API that the zone owner uses to import the ZSK sets of the other providers into their DNSKEY RRset.
- o The DNSKEY RRset is signed independently by each provider using their own KSK.
- o The zone owner manages the DS RRset located in the parent zone. This is comprised of DS records corresponding to the KSKs of each provider.
- o Key rollovers need coordinated participation of the zone owner to update the DS RRset (for KSK), and the DNSKEY RRset (for ZSK).

3. Validating Resolver Behavior

The central requirement for both of the Multiple Signer models (Section 2.1) is to ensure that the ZSKs from all providers are present in each provider's apex DNSKEY RRset, and is vouched for by either the single KSK (in model 1) or each provider's KSK (in model 2.) If this is not done, the following situation can arise (assuming two providers A and B):

- o The validating resolver follows a referral (i.e. secure delegation) to the zone in question.
- o It retrieves the zone's DNSKEY RRset from one of provider A's nameservers, authenticates it against the parent DS RRset, and caches it.
- o At some point in time, the resolver attempts to resolve a name in the zone, while the DNSKEY RRset received from provider A is still viable in its cache.
- o It queries one of provider B's nameservers to resolve the name, and obtains a response that is signed by provider B's ZSK, which it cannot authenticate because this ZSK is not present in its cached DNSKEY RRset for the zone that it received from provider A.

- o The resolver will not accept this response. It may still be able to ultimately authenticate the name by querying other nameservers for the zone until it elicits a response from one of provider A's nameservers. But it has incurred the penalty of additional roundtrips with other nameservers, with the corresponding latency and processing costs. The exact number of additional roundtrips depends on details of the resolver's nameserver selection algorithm and the number of nameservers configured at provider B.
- o It may also be the case that a resolver is unable to provide an authenticated response because it gave up after a certain number of retries or a certain amount of delay, or that downstream clients of the resolver that originated the query timed out waiting for a response.

Hence, it is important that the DNSKEY RRset at each provider is maintained with the active ZSKs of all participating providers. This ensures that resolvers can validate a response no matter which provider's nameservers it came from.

Details of how the DNSKEY RRset itself is validated differ. In model 1 (Section 2.1.1), one unique KSK managed by the zone owner signs an identical DNSKEY RRset deployed at each provider, and the signed DS record in the parent zone refers to this KSK. In model 2 (Section 2.1.2), each provider has a distinct KSK and signs the DNSKEY RRset with it. The zone owner deploys a DS RRset at the parent zone that contains multiple DS records, each referring to a distinct provider's KSK. Hence it does not matter which provider's nameservers the resolver obtains the DNSKEY RRset from, the signed DS record in each model can authenticate the associated KSK.

4. Signing Algorithm Considerations

DNS providers participating in multi-signer models need to use a common DNSSEC signing algorithm (or a common set of algorithms if multiple are in use). This is because the current specifications require that if there are multiple algorithms in the DNSKEY RRset, then RRsets in the zone need to be signed with at least one DNSKEY of each algorithm, as described in RFC 4035 [RFC4035], Section 2.2. If providers employ distinct signing algorithms, then this requirement cannot be satisfied.

5. Authenticated Denial Considerations

Authenticated denial of existence enables a resolver to validate that a record does not exist. For this purpose, an authoritative server presents, in a response to the resolver, signed NSEC (Section 3.1.3 of [RFC4035]) or NSEC3 (Section 7.2 of [RFC5155]) records that

provide cryptographic proof of this non-existence. The NSEC3 method enhances NSEC by providing opt-out for signing insecure delegations and also adds limited protection against zone enumeration attacks.

An authoritative server response carrying records for authenticated denial is always self-contained and the receiving resolver doesn't need to send additional queries to complete the proof of denial. For this reason, no rollover is needed when switching between NSEC and NSEC3 for a signed zone.

Since authenticated denial responses are self-contained, NSEC and NSEC3 can be used by different providers to serve the same zone. Doing so however defeats the protection against zone enumeration provided by NSEC3 (because an adversary can trivially enumerate the zone by just querying the providers that employ NSEC). A better configuration involves multiple providers using different authenticated denial of existence mechanisms that all provide zone enumeration defense, such as pre-computed NSEC3, NSEC3 White Lies [RFC7129], NSEC Black Lies [BLACKLIES], etc. Note however that having multiple providers offering different authenticated denial mechanisms may impact how effectively resolvers are able to make use of the caching of negative responses.

5.1. Single Method

Usually, the NSEC and NSEC3 methods are used exclusively (i.e. the methods are not used at the same time by different servers). This configuration is preferred because the behavior is well-defined and is closest to current operational practice.

5.2. Mixing Methods

Compliant resolvers should be able to validate zone data when different authoritative servers for the same zone respond with different authenticated denial methods because this is normally observed when NSEC and NSEC3 are being switched or when NSEC3PARAM is updated.

Resolver software may, however, be designed to handle a single transition between two authenticated denial configurations more optimally than a permanent setup with mixed authenticated denial methods. This could make caching on the resolver side less efficient and the authoritative servers may observe higher number of queries. This aspect should be considered especially in the context of Aggressive Use of DNSSEC-Validated Cache [RFC8198].

In case all providers cannot be configured with the same authenticated denial mechanism, it is recommended to limit the distinct configurations to the lowest number feasible.

Note that NSEC3 configuration on all providers with different NSEC3PARAM values is considered a mixed setup.

6. Key Rollover Considerations

The Multiple Signer (Section 2.1) models introduce some new requirements for DNSSEC key rollovers. Since this process necessarily involves coordinated actions on the part of providers and the zone owner, one reasonable strategy is for the zone owner to initiate key rollover operations. But other operationally plausible models may also suit, such as a DNS provider initiating a key rollover and signaling their intent to the zone owner in some manner. The mechanism to communicate this intent could be some secure out-of-band channel that has been agreed upon, or the provider could offer an API function that could be periodically polled by the zone owner.

The descriptions in this section assume two DNS providers for simplicity. They also assume that KSK rollovers employ the commonly used Double Signature KSK Rollover Method, and that ZSK rollovers employ the Pre-Publish ZSK Rollover Method, as described in detail in [RFC6781]. With minor modifications, they can be easily adapted to other models, such as Double DS KSK Rollover or Double Signature ZSK rollover, if desired. Key use timing should follow the recommendations outlined in [RFC6781], but taking into account the additional operations needed by the multi signer models. For example, "time to propagate data to all the authoritative servers" now includes the time to import the new ZSKs into each provider.

6.1. Model 1: Common KSK, Unique ZSK per provider

- o Key Signing Key Rollover: In this model, the two managed DNS providers share a common KSK (public key) in their respective zones, and the zone owner controls the KSK signing key. To initiate the rollover, the zone owner generates a new KSK and obtains the DNSKEY RRset of each DNS provider using their respective APIs. The new KSK is added to each provider's DNSKEY RRset and the RRset is re-signed with both the new and the old KSK. This new DNSKEY RRset is then transferred to each provider. The zone owner then updates the DS RRset in the parent zone to point to the new KSK, and after the necessary DS record TTL period has expired, proceeds with updating the DNSKEY RRset to remove the old KSK.

- o **Zone Signing Key Rollover:** In this model, each DNS provider has separate Zone Signing Keys. Each provider can choose to roll their ZSK independently by co-ordinating with the zone owner. Provider A would generate a new ZSK and communicate their intent to perform a rollover (note that Provider A cannot immediately insert this new ZSK into their DNSKEY RRset because the RRset has to be signed by the zone owner). The zone owner obtains the new ZSK from Provider A. It then obtains the current DNSKEY RRset from each provider (including Provider A), inserts the new ZSK into each DNSKEY RRset, re-signs the DNSKEY RRset, and sends it back to each provider for deployment via their respective key management APIs. Once the necessary time period is elapsed (i.e. all zone data has been re-signed by the new ZSK and propagated to all authoritative servers for the zone, plus the maximum zone TTL value of any of the data in the zone signed by the old ZSK), Provider A and the zone owner can initiate the next phase of removing the old ZSK, and re-signing the resulting new DNSKEY RRset.

6.2. Model 2: Unique KSK and ZSK per provider

- o **Key Signing Key Rollover:** In Model 2, each managed DNS provider has their own KSK. A KSK roll for provider A does not require any change in the DNSKEY RRset of provider B, but does require co-ordination with the zone owner in order to get the DS record set in the parent zone updated. The KSK roll starts with Provider A generating a new KSK and including it in their DNSKEY RRset. The DNSKEY RRset would then be signed by both the new and old KSK. The new KSK is communicated to the zone owner, after which the zone owner updates the DS RRset to replace the DS record for the old KSK with a DS record for the new KSK. After the necessary DS RRset TTL period has elapsed, the old KSK can be removed from provider A's DNSKEY RRset.
- o **Zone Signing Key Rollover:** In Model 2, each managed DNS provider has their own ZSK. The ZSK roll for provider A would start with them generating new ZSK and including it in their DNSKEY RRset and re-signing the new DNSKEY RRset with their KSK. The new ZSK of provider A would then be communicated to the zone owner, who will initiate the process of importing this ZSK into the DNSKEY RRsets of the other providers, using their respective APIs. Once the necessary Pre-Publish key rollover time periods have elapsed, provider A and the zone owner can initiate the process of removing the old ZSK from the DNSKEY RRset of all providers.

7. Using Combined Signing Keys

A Combined Signing Key (CSK) is one in which the same key serves the purpose of being both the secure entry point (SEP) key for the zone, and also for signing all the zone data including the DNSKEY RRset (i.e., there is no KSK/ZSK split).

Model 1 is not compatible with CSKs because the zone owner would then hold the sole signing key, and providers would not be able to sign their own zone data.

Model 2 can accommodate CSKs without issue. In this case, any or all of the providers could employ a CSK. The DS record in the parent zone would reference the provider's CSK instead of KSK, and the public CSK will need to be imported into the DNSKEY RRsets of all of the other providers. A CSK key rollover for such a provider would involve the following: The provider generates a new CSK, installs the new CSK into the DNSKEY RRset, and signs it with both the old and new CSK. The new CSK is communicated to the zone owner. The zone owner exports this CSK into the other provider's DNSKEY RRsets and replaces the DS record referencing the old CSK with one referencing the new one in the parent DS RRset. Once all the zone data has been re-signed with the new CSK, the old CSK is removed from the DNSKEY RRset, and the latter is re-signed with only the new CSK. Finally, the old CSK is removed from the DNSKEY RRsets of the other providers.

8. Use of CDS and CDNSKEY

CDS and CDNSKEY records [RFC7344] [RFC8078] are used to facilitate automated updates of DNSSEC secure entry point keys between parent and child zones. Multi-signer DNSSEC configurations can support this too. In Model 1, CDS/CDNSKEY changes are centralized at the zone owner. However, the zone owner will still need to push down updated signed CDNS/DNSKEY RRsets to the providers via the key management mechanism. In Model 2, the key management mechanism needs to support cross importation of the CDS/CDNSKEY records, so that a common view of the RRset can be constructed at each provider, and is visible to the parent zone attempting to update the DS RRset.

9. Key Management Mechanism Requirements

Managed DNS providers typically have their own proprietary zone configuration and data management APIs, commonly utilizing HTTPS/REST interfaces. So, rather than outlining a new API for key management here, we describe the specific functions that the provider API needs to support in order to enable the multi-signer models. The zone owner is expected to use these API functions to perform key management tasks. Other mechanisms that can partly offer these

functions, if supported by the providers, include the DNS UPDATE protocol [RFC2136] and EPP [RFC5731].

- o The API must offer a way to query the current DNSKEY RRset of the provider
- o For model 1, the API must offer a way to import a signed DNSKEY RRset and replace the current one at the provider. Additionally, if CDS/CDNSKEY is supported, the API must also offer a way to import a signed CDS/CDNSKEY RRset.
- o For model 2, the API must offer a way to import a DNSKEY record from an external provider into the current DNSKEY RRset. Additionally, if CDS/CDNSKEY is supported, the API must offer a mechanism to import individual CDS/CDNSKEY records from an external provider.

In model 2, once initially bootstrapped with each other's zone signing keys via these API mechanisms, providers could, if desired, periodically query each other's DNSKEY RRsets, authenticate their signatures, and automatically import or withdraw ZSKs in the keyset as key rollover events happen.

10. DNS Response Size Considerations

The Multi-Signer models result in larger DNSKEY RRsets, so the size of a response to a query for the DNSKEY RRset will be larger. The actual size increase depends on multiple factors: DNSKEY algorithm and keysize choices, the number of providers, whether additional keys are pre-published, how many simultaneous key rollovers are in progress etc. Newer elliptic curve algorithms produce keys small enough that the responses will typically be far below the common Internet path MTU. Thus operational concerns related to IP fragmentation or truncation and TCP fallback are unlikely to be encountered. In any case, DNS operators need to ensure that they can emit and process large DNS UDP responses when necessary, and a future migration to alternative transports like DNS over TLS [RFC7858] or DNS over HTTPS [RFC8484] may make this topic moot.

11. IANA Considerations

This document includes no request to IANA.

12. Security Considerations

The Multi Signer models necessarily involve 3rd party providers holding the private keys that sign the zone owner's data. Obviously this means that the zone owner has decided to place a great deal of

trust in these providers. By contrast, the more traditional model in which the zone owner runs a hidden master and uses the zone transfer protocol with the providers, is arguably more secure, because only the zone owner holds the private signing keys, and the 3rd party providers cannot serve bogus data without detection by validating resolvers.

The Zone key import and export APIs required by these models need to be strongly authenticated to prevent tampering of key material by malicious third parties. Many providers today offer REST/HTTPS APIs, where the HTTPS layer provides transport security and server authentication, and that utilize a number of client authentication mechanisms (username/password, API keys etc). If DNS protocol mechanisms like UPDATE are being used for key insertion and deletion, they should similarly be strongly authenticated, e.g. by employing Transaction Signatures (TSIG) [RFC2845]. Multi-factor authentication could be used to further strengthen security. Key Generation and other general security related operations should follow the guidance specified in [RFC6781].

13. Acknowledgments

The initial version of this document benefited from discussions with and review from Duane Wessels. Additional helpful comments were provided by Steve Crocker, Ulrich Wisser, Tony Finch, Olafur Gudmundsson, Matthijs Mekking, Daniel Migault, and Ben Kaduk.

14. References

14.1. 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>>.
- [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>>.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.

- [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>>.
- [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>>.
- [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>>.
- [RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <<https://www.rfc-editor.org/info/rfc6781>>.
- [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>>.
- [RFC8078] Gudmundsson, O. and P. Wouters, "Managing DS Records from the Parent via CDS/CDNSKEY", RFC 8078, DOI 10.17487/RFC8078, March 2017, <<https://www.rfc-editor.org/info/rfc8078>>.
- [RFC8198] Fujiwara, K., Kato, A., and W. Kumari, "Aggressive Use of DNSSEC-Validated Cache", RFC 8198, DOI 10.17487/RFC8198, July 2017, <<https://www.rfc-editor.org/info/rfc8198>>.

14.2. Informative References

- [BLACKLIES] Valsorda, F. and O. Gudmundsson, "Compact DNSSEC Denial of Existence or Black Lies", <<https://tools.ietf.org/html/draft-valsorda-dnsop-black-lies>>.
- [RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, DOI 10.17487/RFC1995, August 1996, <<https://www.rfc-editor.org/info/rfc1995>>.

- [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>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<https://www.rfc-editor.org/info/rfc5731>>.
- [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>>.
- [RFC7129] Gieben, R. and W. Mekking, "Authenticated Denial of Existence in the DNS", RFC 7129, DOI 10.17487/RFC7129, February 2014, <<https://www.rfc-editor.org/info/rfc7129>>.
- [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>>.
- [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>>.

Authors' Addresses

Shumon Huque
Salesforce

Email: shuque@gmail.com

Pallavi Aras
Salesforce

Email: paras@salesforce.com

John Dickinson
Sinodun

Email: jad@sinodun.com

Jan Vcelak
NS1

Email: jvcelak@ns1.com

David Blacka
Verisign

Email: davidb@verisign.com

DNSOP Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 29, 2019

L. Lhotka
P. Spacek
CZ.NIC
June 27, 2019

YANG Types for DNS Classes and Resource Record Types
draft-lhotka-dnsop-iana-class-type-yang-02

Abstract

This document contains the initial revision of the YANG module `iana-dns-class-rr-type` that contains derived types reflecting two IANA registries: DNS CLASSES and Resource Record (RR) TYPES. These YANG types are intended as a minimum basis for future data modeling work.

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 December 29, 2019.

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
2. YANG Design Considerations	3
3. YANG Module	4
4. IANA Considerations	22
4.1. URI Registrations	23
4.2. YANG Module Registrations	23
5. Security Considerations	24
6. References	24
6.1. Normative References	24
6.2. Informative References	24
Authors' Addresses	24

1. Introduction

YANG [RFC7950] has become a de facto standard as a language for modeling configuration and state data, as well as specifying management operations and asynchronous notifications. It is reasonable to expect that the approach based on utilizing such data models along with standard management protocols such as NETCONF and RESTCONF can be effectively used in DNS operations, too. In fact, several efforts are currently underway that attempt to use NETCONF or RESTCONF for configuring and managing

- o authoritative servers
- o resolvers
- o zone data.

While it is possible to use the management protocols mentioned above with ad hoc or proprietary data models, their real potential can be realized only if there is a (completely or partly) unified data model supported by multiple DNS software implementations. Operators can then, for instance, run several different DNS servers in parallel, and use a common configuration and management interface and data for all of them. Also, it becomes considerably easier to migrate to another implementation.

Based on the previous experience from the IETF Routing Area, it is to be expected that the development of unified data models for DNS will be a lengthy and complicated process that will require active cooperation and compromises from the vendors and developers of major DNS server platforms. Nevertheless, it is likely that any DNS-related data modeling effort will need to use various DNS parameters and enumerations that are specified in several IANA registries. For use with YANG, these parameters and enumerations have to be

translated into corresponding YANG types or other structures. Such translations should be straightforward and relatively uncontroversial.

This document is a first step in translating DNS-related IANA registries to YANG. It contains the initial revision of the YANG module "iana-dns-class-rr-type" that defines derived types for the common parameters of DNS resource records (RR): class and type. These YANG types, "dns-class" and "rr-type", reflect the IANA registries "DNS CLASSES" and "Resource Record (RR) TYPES" [IANA-DNS-PARAMETERS].

It is worth emphasizing that the role of the DNSOP Working Group is only in preparing and publishing this initial revision of the YANG module. Subsequently, whenever a new class or RR type is added to the above registries, IANA will also update the iana-dns-class-rr-type YANG module, following the instructions in Section 4 below.

2. YANG Design Considerations

The IANA document "Domain Name System (DNS) Parameters" [IANA-DNS-PARAMETERS] contains altogether thirteen registries. The YANG module iana-dns-class-rr-type defines derived types corresponding to only two of the registries that are essential for data models involving zone data, namely "DNS CLASSES" and "Resource Record (RR) TYPES". It is expected that the remaining registries in [IANA-DNS-PARAMETERS], as well as other DNS-related IANA registries, will be analogically reflected in future YANG modules as necessary. This way, an appropriate combination of YANG modules can be chosen depending on which YANG types are needed for a given data modeling purpose.

[RFC3597] introduced the option of specifying a class or type via its assigned decimal number, as an alternative to the mnemonic name. For example, the "IN" class can be equivalently written as "CLASS1", and "AAAA" type as "TYPE28".

Accordingly, the derived types "dns-class" and "rr-type" are defined in the YANG module as a union of two member types:

- o 16-bit decimal integer ("uint16")
- o mnemonic name, represented by the enumeration type "dns-class-name" and "rr-type-name", respectively.

As unassigned and reserved class and types values are not included in the mnemonic name enumerations, they can be used only via their decimal codes.

3. YANG Module

RFC Editor: In this section, replace all occurrences of "XXXX" with the actual RFC number and all occurrences of the revision date below with the date of RFC publication (and remove this note).

```
<CODE BEGINS> file "ietf-iana-dns-class-rr-type@2019-06-27.yang"
```

```
module iana-dns-class-rr-type {  
    yang-version 1.1;  
  
    namespace "urn:ietf:params:xml:ns:yang:iana-dns-class-rr-type";  
    prefix dnsct;  
  
    organization  
        "Internet Assigned Numbers Authority (IANA)";  
  
    contact  
        "  
            Internet Assigned Numbers Authority  
  
        Postal: ICANN  
            4676 Admiralty Way, Suite 330  
            Marina del Rey, CA 90292  
  
        Tel:      +1 310 823 9358  
  
        <mailto:iana@iana.org>";  
  
    description  
        "This YANG module translates IANA registries 'DNS CLASSes' and  
        'Resource Record (RR) TYPES' to YANG derived types.  
  
        Copyright (c) 2018 IETF Trust and the persons identified as  
        authors of the code. All rights reserved.  
  
        Redistribution and use in source and binary forms, with or  
        without modification, is permitted pursuant to, and subject to  
        the license terms contained in, the Simplified BSD License set  
        forth in Section 4.c of the IETF Trust's Legal Provisions  
        Relating to IETF Documents  
        (https://trustee.ietf.org/license-info).  
  
        This version of this YANG module is part of RFC XXXX  
        (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for  
        full legal notices.";
```

```
reference
  "IANA 'Domain Name System (DNS) Parameters' registry
  https://www.iana.org/assignments/dns-parameters";

revision 2019-06-27 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Types for DNS Classes and Resource Record
    Types";
}

/* Typedefs */

typedef dns-class-name {
  type enumeration {
    enum IN {
      value "1";
      description
        "Internet";
      reference
        "RFC 1035: Domain Names - Implementation and
        Specification";
    }
    enum CH {
      value "3";
      description
        "Chaos";
      reference
        "Moon, D., 'Chaosnet', A. I. Memo 628, MIT Artificial
        Intelligence Laboratory, June 1981";
    }
    enum HS {
      value "4";
      description
        "Hesiod";
      reference
        "Dyer, S. and Hsu, F, 'Hesiod', Project Athena Technical
        Plan - Name Service, April 1987";
    }
    enum NONE {
      value "254";
      description
        "QCLASS NONE";
      reference
        "RFC 2136: Dynamic Updates in the Domain Name System (DNS
        UPDATE)";
    }
  }
}
```

```
    enum ANY {
        value "255";
        description
            "QCLASS * (ANY)";
        reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
    }
}
description
    "This enumeration type defines mnemonic names and corresponding
     numeric values of DNS classes.";
reference
    "RFC 6895: Domain Name System (DNS) IANA Considerations";
}

typedef dns-class {
    type union {
        type uint16;
        type dns-class-name;
    }
    description
        "This type allows for referring to a DNS class using either the
         assigned mnemonic name or numeric value.";
}

typedef rr-type-name {
    type enumeration {
        enum A {
            value "1";
            description
                "A host address.";
            reference
                "RFC 1035: Domain Names - Implementation and
                 Specification";
        }
        enum NS {
            value "2";
            description
                "An authoritative name server.";
            reference
                "RFC 1035: Domain Names - Implementation and
                 Specification";
        }
        enum MD {
            value "3";
            status "obsolete";
            description
```

```
        "A mail destination (obsolete - use MX).";
    reference
        "RFC 1035: Domain Names - Implementation and
        Specification";
}
enum MF {
    value "4";
    status "obsolete";
    description
        "A mail forwarder (obsolete - use MX).";
    reference
        "RFC 1035: Domain Names - Implementation and
        Specification";
}
enum CNAME {
    value "5";
    description
        "The canonical name for an alias.";
    reference
        "RFC 1035: Domain Names - Implementation and
        Specification";
}
enum SOA {
    value "6";
    description
        "Start of a zone of authority.";
    reference
        "RFC 1035: Domain Names - Implementation and
        Specification";
}
enum MB {
    value "7";
    description
        "A mailbox domain name (experimental).";
    reference
        "RFC 1035: Domain Names - Implementation and
        Specification";
}
enum MG {
    value "8";
    description
        "A mail group member (experimental).";
    reference
        "RFC 1035: Domain Names - Implementation and
        Specification";
}
enum MR {
    value "9";
```

```
    description
      "A mail rename domain name (experimental).";
    reference
      "RFC 1035: Domain Names - Implementation and
        Specification";
  }
  enum NULL {
    value "10";
    description
      "A null RR (experimental).";
    reference
      "RFC 1035: Domain Names - Implementation and
        Specification";
  }
  enum WKS {
    value "11";
    description
      "A well known service description.";
    reference
      "RFC 1035: Domain Names - Implementation and
        Specification";
  }
  enum PTR {
    value "12";
    description
      "A domain name pointer.";
    reference
      "RFC 1035: Domain Names - Implementation and
        Specification";
  }
  enum HINFO {
    value "13";
    description
      "Host information.";
    reference
      "RFC 1035: Domain Names - Implementation and
        Specification";
  }
  enum MINFO {
    value "14";
    description
      "Mailbox or mail list information.";
    reference
      "RFC 1035: Domain Names - Implementation and
        Specification";
  }
  enum MX {
    value "15";
```



```
    description
      "Mail exchange.";
    reference
      "RFC 1035: Domain Names - Implementation and
      Specification";
  }
  enum TXT {
    value "16";
    description
      "Text strings.";
    reference
      "RFC 1035: Domain Names - Implementation and
      Specification";
  }
  enum RP {
    value "17";
    description
      "Responsible person.";
    reference
      "RFC 1183: New DNS RR Definitions";
  }
  enum AFSDDB {
    value "18";
    description
      "AFS data base location.";
    reference
      "- RFC 1183: New DNS RR Definitions
      - RFC 5864: DNS SRV Resource Records for AFS";
  }
  enum X25 {
    value "19";
    description
      "X.25 PSDN address.";
    reference
      "RFC 1183: New DNS RR Definitions";
  }
  enum ISDN {
    value "20";
    description
      "ISDN address.";
    reference
      "RFC 1183: New DNS RR Definitions";
  }
  enum RT {
    value "21";
    description
      "Route through.";
```

```
    reference
      "RFC 1183: New DNS RR Definitions";
  }
  enum NSAP {
    value "22";
    description
      "NSAP address, NSAP style A record.";
    reference
      "RFC 1706: DNS NSAP Resource Records";
  }
  enum NSAP-PTR {
    value "23";
    description
      "Domain name pointer, NSAP style.";
    reference
      "- RFC 1348: DNS NSAP RRs

      - RFC 1637: DNS NSAP Resource Records

      - RFC 1706: DNS NSAP Resource Records";
  }
  enum SIG {
    value "24";
    description
      "Security signature.";
    reference
      "- RFC 4034: Resource Records for the DNS Security
        Extensions

      - RFC 3755: Legacy Resolver Compatibility for Delegation
        Signer (DS)

      - RFC 2535: Domain Name System Security Extensions

      - RFC 2536: DSA KEYs and SIGs in the Domain Name System
        (DNS)

      - RFC 2537: RSA/MD5 KEYs and SIGs in the Domain Name
        System (DNS)

      - RFC 2931: DNS Request and Transaction Signatures
        (SIG(0)s)

      - RFC 3110: RSA/SHA-1 SIGs and RSA KEYs in the Domain Name
        System (DNS)

      - RFC 3008: Domain Name System Security (DNSSEC) Signing
        Authority";
```

```
}
enum KEY {
  value "25";
  description
    "Security key.";
  reference
    "- RFC 4034: Resource Records for the DNS Security
      Extensions

    - RFC 3755: Legacy Resolver Compatibility for Delegation
      Signer (DS)

    - RFC 2535: Domain Name System Security Extensions

    - RFC 2536: DSA KEYs and SIGs in the Domain Name System
      (DNS)

    - RFC 2537: RSA/MD5 KEYs and SIGs in the Domain Name
      System (DNS)

    - RFC 2539: Storage of Diffie-Hellman Keys in the Domain
      Name System (DNS)

    - RFC 3008: Domain Name System Security (DNSSEC) Signing
      Authority

    - RFC 3110: RSA/SHA-1 SIGs and RSA KEYs in the Domain Name
      System (DNS)";
}
enum PX {
  value "26";
  description
    "X.400 mail mapping information.";
  reference
    "RFC 2163: Using the Internet DNS to Distribute MIXER
      Conformant Global Address Mapping (MCGAM)";
}
enum GPOS {
  value "27";
  description
    "Geographical position.";
  reference
    "RFC 1712: DNS Encoding of Geographical Location";
}
enum AAAA {
  value "28";
  description
    "IPv6 address.";
```

```
        reference
            "RFC 3596: DNS Extensions to Support IP Version 6";
    }
    enum LOC {
        value "29";
        description
            "Location information.";
        reference
            "RFC 1876: A Means for Expressing Location Information in
            the Domain Name System";
    }
    enum NXT {
        value "30";
        status "obsolete";
        description
            "Next domain (obsolete).";
        reference
            "- RFC 3755: Legacy Resolver Compatibility for Delegation
            Signer (DS)

            - RFC 2535: Domain Name System Security Extensions";
    }
    enum EID {
        value "31";
        description
            "Endpoint identifier.";
    }
    enum NIMLOC {
        value "32";
        description
            "Nimrod locator.";
    }
    enum SRV {
        value "33";
        description
            "Server selection.";
        reference
            "RFC 2782: A DNS RR for specifying the location of services
            (DNS SRV)";
    }
    enum ATMA {
        value "34";
        description
            "ATM address.";
        reference
            "ATM Forum Technical Committee, 'ATM Name System V2.0',
            AF-DANS-0152.00, July 2000";
    }
}
```

```
enum NAPTR {
  value "35";
  description
    "Naming authority pointer.";
  reference
    "- RFC 2915: The Naming Authority Pointer (NAPTR) DNS
      Resource Record

      - RFC 2168: Resolution of Uniform Resource Identifiers
        using the Domain Name System

      - RFC 3403: Dynamic Delegation Discovery System (DDDS)
        Part Three: The Domain Name System (DNS) Database";
}
enum KX {
  value "36";
  description
    "Key exchanger.";
  reference
    "RFC 2230: Key Exchange Delegation Record for the DNS";
}
enum CERT {
  value "37";
  description
    "Certificate.";
  reference
    "RFC 4398: Storing Certificates in the Domain Name System
      (DNS) ";
}
enum A6 {
  value "38";
  status "obsolete";
  description
    "IPv6 address (obsolete - use AAAA).";
  reference
    "- RFC 3226: DNSSEC and IPv6 A6 Aware Server/Resolver
      Message Size Requirements

      - RFC 2874: DNS Extensions to Support IPv6 Address
        Aggregation and Renumbering

      - RFC 6563: Moving A6 to Historic Status";
}
enum DNAME {
  value "39";
  description
    "DNAME.";
  reference
```

```
    "- RFC 2672: Non-Terminal DNS Name Redirection
    - RFC 6672: DNAME Redirection in the DNS";
}
enum SINK {
    value "40";
    description
        "Kitchen sink.";
}
enum OPT {
    value "41";
    description
        "OPT pseudo-RR.";
    reference
        "- RFC 6891: Extension Mechanisms for DNS (EDNS(0))
        - RFC 3225: Indicating Resolver Support of DNSSEC";
}
enum APL {
    value "42";
    description
        "Address prefix list.";
    reference
        "RFC 3123: A DNS RR Type for Lists of Address Prefixes (APL
        RR)";
}
enum DS {
    value "43";
    description
        "Delegation signer.";
    reference
        "- RFC 4034: Resource Records for the DNS Security
        Extensions
        - RFC 3658: Delegation Signer (DS) Resource Record (RR)";
}
enum SSHFP {
    value "44";
    description
        "SSH key fingerprint.";
    reference
        "RFC 4255: Using DNS to Securely Publish Secure Shell (SSH)
        Key Fingerprints";
}
enum IPSECKEY {
    value "45";
    description
        "IPSec key.";
```

```
        reference
        "RFC 4025: A Method for Storing IPsec Keying Material in
        DNS";
    }
    enum RRSIG {
        value "46";
        description
        "RR signature.";
        reference
        "- RFC 4034: Resource Records for the DNS Security
        Extensions

        - RFC 3755: Legacy Resolver Compatibility for Delegation
        Signer (DS) ";
    }
    enum NSEC {
        value "47";
        description
        "NSEC resource record.";
        reference
        "- RFC 4034: Resource Records for the DNS Security
        Extensions

        - RFC 3755: Legacy Resolver Compatibility for Delegation
        Signer (DS) ";
    }
    enum DNSKEY {
        value "48";
        description
        "DNSKEY resource record.";
        reference
        "- RFC 4034: Resource Records for the DNS Security
        Extensions

        - RFC 3755: Legacy Resolver Compatibility for Delegation
        Signer (DS) ";
    }
    enum DHCID {
        value "49";
        description
        "DHCID resource record.";
        reference
        "RFC 4701: A DNS Resource Record (RR) for Encoding Dynamic
        Host Configuration Protocol (DHCP) Information (DHCID
        RR) ";
    }
    enum NSEC3 {
        value "50";
```

```
    description
        "NSEC3 resource record.";
    reference
        "RFC 5155: DNS Security (DNSSEC) Hashed Authenticated
        Denial of Existence";
}
enum NSEC3PARAM {
    value "51";
    description
        "NSEC3PARAM resource record.";
    reference
        "RFC 5155: DNS Security (DNSSEC) Hashed Authenticated
        Denial of Existence";
}
enum TLSA {
    value "52";
    description
        "TLSA resource record.";
    reference
        "RFC 6698: The DNS-Based Authentication of Named Entities
        (DANE) Transport Layer Security (TLS) Protocol: TLSA";
}
enum SMIMEA {
    value "53";
    description
        "S/MIME cert association";
    reference
        "RFC 8162: Using Secure DNS to Associate Certificates with
        Domain Names for S/MIME";
}
enum HIP {
    value "55";
    description
        "Host identity protocol.";
    reference
        "RFC 5205: Host Identity Protocol (HIP) Domain Name System
        (DNS) Extension";
}
enum NINFO {
    value "56";
    description
        "NINFO resource record.";
}
enum RKEY {
    value "57";
    description
        "RKEY resource record.";
}
```



```
enum TALINK {
  value "58";
  description
    "Trust anchor LINK.";
}
enum CDS {
  value "59";
  description
    "Child DS.";
  reference
    "RFC 7344: Automating DNSSEC Delegation Trust
    Maintenance";
}
enum CDNSKEY {
  value "60";
  description
    "DNSKEY(s) the child wants reflected in DS.";
  reference
    "RFC 7344: Automating DNSSEC Delegation Trust
    Maintenance";
}
enum OPENPGPKEY {
  value "61";
  description
    "OpenPGP key.";
  reference
    "RFC 7929: DNS-Based Authentication of Named Entities
    (DANE) Bindings for OpenPGP";
}
enum CSYNC {
  value "62";
  description
    "Child-to-parent synchronization.";
  reference
    "RFC 7477: Child-to-Parent Synchronization in DNS";
}
enum SPF {
  value "99";
  description
    "SPF (sender policy framework) resource record.";
  reference
    "RFC 7208: Sender Policy Framework (SPF) for Authorizing
    Use of Domains in Email, Version 1";
}
enum UINFO {
  value "100";
  description
    "IANA-reserved.";
```

```
}
enum UID {
  value "101";
  description
    "IANA-reserved.";
}
enum GID {
  value "102";
  description
    "IANA-reserved.";
}
enum UNSPEC {
  value "103";
  description
    "IANA-reserved.";
}
enum NID {
  value "104";
  description
    "Node identifier.";
  reference
    "RFC 6742: DNS Resource Records for the Identifier-Locator
    Network Protocol (ILNP)";
}
enum L32 {
  value "105";
  description
    "L32 resource record.";
  reference
    "RFC 6742: DNS Resource Records for the Identifier-Locator
    Network Protocol (ILNP)";
}
enum L64 {
  value "106";
  description
    "L64 resource record.";
  reference
    "RFC 6742: DNS Resource Records for the Identifier-Locator
    Network Protocol (ILNP)";
}
enum LP {
  value "107";
  description
    "LP resource record.";
  reference
    "RFC 6742: DNS Resource Records for the Identifier-Locator
    Network Protocol (ILNP)";
}
```

```
enum EUI48 {
  value "108";
  description
    "An EUI-48 address.";
  reference
    "RFC 7043: Resource Records for EUI-48 and EUI-64 Addresses
    in the DNS";
}
enum EUI64 {
  value "109";
  description
    "An EUI-64 address.";
  reference
    "RFC 7043: Resource Records for EUI-48 and EUI-64 Addresses
    in the DNS";
}
enum TKEY {
  value "249";
  description
    "Transaction key.";
  reference
    "RFC 2930: Secret Key Establishment for DNS (TKEY RR)";
}
enum TSIG {
  value "250";
  description
    "Transaction signature.";
  reference
    "RFC 2845: Secret Key Transaction Authentication for DNS
    (TSIG)";
}
enum IXFR {
  value "251";
  description
    "Incremental transfer.";
  reference
    "RFC 1995: Incremental Zone Transfer in DNS";
}
enum AXFR {
  value "252";
  description
    "Transfer of an entire zone.";
  reference
    "- RFC 1035: Domain Names - Implementation and
    Specification

    - RFC 5936: DNS Zone Transfer Protocol (AXFR)";
}
```

```
enum MAILB {
  value "253";
  description
    "Mailbox-related RRs (MB, MG or MR).";
  reference
    "RFC 1035: Domain Names - Implementation and
    Specification";
}
enum MAILA {
  value "254";
  status "obsolete";
  description
    "Mail agent RRs (obsolete - see MX).";
  reference
    "RFC 1035: Domain Names - Implementation and
    Specification";
}
enum * {
  value "255";
  description
    "A request for all records the server/cache has
    available.";
  reference
    "- RFC 1035: Domain Names - Implementation and
    Specification

    - RFC 6895: Domain Name System (DNS) IANA
    Considerations";
}
enum URI {
  value "256";
  description
    "URI resource record.";
  reference
    "RFC 7553: The Uniform Resource Identifier (URI) DNS
    Resource Record";
}
enum CAA {
  value "257";
  description
    "Certification authority authorization.";
  reference
    "RFC 6844: DNS Certification Authority Authorization (CAA)
    Resource Record";
}
enum AVC {
  value "258";
  description
```

```
        "Application visibility and control.";
    }
    enum DOA {
        value "259";
        description
            "Digital object architecture";
        reference
            "draft-durand-doa-over-dns: DOA over DNS";
    }
    enum AMTRELAY {
        value "260";
        description
            "Automatic multicast tunneling relay.";
        reference
            "J. Holland: DNS Reverse IP AMT Discovery.
             draft-ietf-mboned-driad-amt-discovery.";
    }
    enum TA {
        value "32768";
        description
            "DNSSEC trust authorities.";
    }
    enum DLV {
        value "32769";
        description
            "DNSSEC lookaside validation.";
        reference
            "RFC 4431: The DNSSEC Lookaside Validation (DLV) DNS
             Resource Record";
    }
}
description
    "This enumeration type defines mnemonic names and corresponding
     numeric values of DNS resource record types.";
reference
    "- RFC 6895: Domain Name System (DNS) IANA Considerations
     - RFC 1035: Domain Names - Implementation and Specification";
}

typedef rr-type {
    type union {
        type uint16;
        type rr-type-name;
    }
    description
        "This type allows for referring to a DNS resource record type
         using either the assigned mnemonic name or numeric value.";
```

```
    }  
  }
```

<CODE ENDS>

4. IANA Considerations

RFC Editor: In this section, replace all occurrences of "XXXX" with the actual RFC number (and remove this note).

This document defines the initial version of the IANA-maintained iana-dns-class-rr-type YANG module.

The iana-dns-class-rr-type YANG module is intended to reflect the "DNS CLASSes" and "Resource Record (RR) TYPEs" registries in [IANA-DNS-PARAMETERS].

IANA has added this new note to the "iana-dns-class-rr-type YANG Module" registry:

Classes and types of DNS resource records must not be directly added to the iana-dns-class-rr-type YANG module. They must instead be added to the "DNS CLASSes" and "Resource Record (RR) TYPEs" registries, respectively.

When a new DNS class or RR type is added to the "DNS CLASSes" or "Resource Record (RR) TYPEs" registry, a new "enum" statement must be added to the "dns-class-name" or "rr-type-name" type, respectively. The assigned name defined by the "enum" statement is the same as the mnemonic name of the new class or type. The following substatements to the "enum" statement should be defined:

"value": Use the decimal value from the registry.

"status": Include only if a class or type registration has been deprecated (use the value "deprecated") or obsoleted (use the value "obsolete").

"description": Replicate the corresponding information from the registry, namely the full name of the new DNS class, or the meaning of the new RR type, if any.

"reference": Replicate the reference from the registry, if any, and add the title of the document, if applicable.

Unassigned or reserved values are not included in the "dns-class-name" and "rr-type-name" enumeration types.

Each time the iana-dns-class-rr-type YANG module is updated, a new "revision" statement must be added before the existing "revision" statements.

IANA has added this new note to the "DNS CLASSes" and "Resource Record (RR) TYPEs" registries:

When this registry is modified, the YANG module iana-dns-class-rr-type must be updated as defined in RFC XXXX.

The "Reference" text in the "DNS CLASSes" registry has been updated as follows:

OLD:
[RFC6895]

NEW:
[RFC6895] [RFCXXXX]

The "Reference" text in the "Resource Record (RR) TYPEs" registry has been updated as follows:

OLD:
[RFC6895] [RFC1035]

NEW:
[RFC6895] [RFC1035] [RFCXXXX]

4.1. URI Registrations

This document registers a URI in the "IETF XML Registry" [RFC3688]. The following registration has been made:

URI: urn:ietf:params:xml:ns:yang:iana-dns-class-rr-type
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

4.2. YANG Module Registrations

This document registers a YANG module in the "YANG Module Names" registry [RFC6020]. The following registration has been made:

name: iana-dns-class-rr-type
namespace: urn:ietf:params:xml:ns:yang:iana-dns-class-rr-type
prefix: dnsct
reference: RFC XXXX

5. Security Considerations

This document translates two IANA registries into YANG data types and otherwise introduces no technology or protocol. Consequently, there are no security issues to be considered for this document.

6. References

6.1. Normative References

- [IANA-DNS-PARAMETERS]
Internet Assigned Numbers Authority, "Domain Name System (DNS) Parameters", January 2018,
<<https://www.iana.org/assignments/dns-parameters>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004,
<<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010,
<<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016,
<<https://www.rfc-editor.org/info/rfc7950>>.

6.2. Informative References

- [RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <<https://www.rfc-editor.org/info/rfc3597>>.

Authors' Addresses

Ladislav Lhotka
CZ.NIC

Email: lhotka@nic.cz

Petr Spacek
CZ.NIC

Email: petr.spacek@nic.cz

Domain Name System Operations
Internet-Draft
Intended status: Informational
Expires: February 14, 2020

J. Livingood
Comcast
August 13, 2019

Responsibility for Authoritative DNS and DNSSEC Mistakes
draft-livingood-dnsop-auth-dnssec-mistakes-05

Abstract

DNS Security Extensions (DNSSEC) validation by recursive DNS resolvers has been deployed at scale. However, domain signing tools and processes are not yet as mature and reliable as is the case for non-DNSSEC-related domain administration tools and processes. This sometimes results in DNSSEC-validation failures, for which operators of validating resolvers are often blamed. This is similar to other, non-DNSSEC-related authoritative DNS errors, for which individual recursive DNS operators are sometimes incorrectly blamed. This document makes clear that responsibility for any and all authoritative DNS failures rests squarely with authoritative domain name operators, who are the only party that can properly maintain their domain names and rectify associated authoritative DNS errors.

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 February 14, 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
2. Domain Validation Failures	3
3. Responsibility for Failures	4
4. Comparison to Other DNS Misconfigurations	5
5. Other Considerations	5
5.1. Security Considerations	5
5.2. Privacy Considerations	5
5.3. IANA Considerations	6
6. Acknowledgements	6
7. References	6
7.1. Normative References	6
7.2. Informative References	7
Appendix A. Document Change Log	7
Appendix B. Open Issues	8
Author's Address	8

1. Introduction

The Domain Name System (DNS), DNS Security Extensions (DNSSEC), and related operational practices are defined extensively [RFC1034] [RFC1035] [RFC4033] [RFC4034] [RFC4035] [RFC4398] [RFC4509] [RFC6781] [RFC5155].

DNS Security Extensions (DNSSEC) validation by recursive DNS resolvers has been deployed at scale. However, domain signing tools and processes are not yet as mature and reliable as is the case for non-DNSSEC-related domain administration tools and processes. This sometimes results in DNSSEC-validation failures, for which operators of validating resolvers are often blamed. This is similar to other, non-DNSSEC-related authoritative DNS errors, for which individual recursive DNS operators are sometimes incorrectly blamed. This documents makes clear that responsibility for any and all authoritative DNS failures rests squarely with authoritative domain name operators, who are the only party that can properly maintain their domain names and rectify associated authoritative DNS errors.

Operators of DNS recursive resolvers, including Internet Service Providers (ISPs) and cloud-based DNS resolvers, occasionally observe

domains incorrectly managing DNSSEC-related resource records. This mismanagement triggers DNSSEC validation failures, and then causes large numbers of end users to be unable to reach a domain. Similarly, errors in non-DNSSEC-related authoritative DNS resource records result in failures, from NXDOMAIN responses to valid responses containing outdated or unreachable hosts.

Many end users, as well as reporters, policymakers, regulators, and others often interpret this as a failure of particular recursive DNS resolvers. Rather than seeing this as a failure on the part of the domain they wanted to reach, they may themselves and/or recommend to others that they switch to a non-validating resolver (which reduces their security), switch to a different DNS resolver (which can reduce non-DNS application layer performance), or contact their ISP or DNS resolver operator to complain.

This document makes clear, however, that responsibility for these types of authoritative DNS failures rests squarely with authoritative domain name operators, as noted in Section 3.

2. Domain Validation Failures

A domain name can fail validation for two general reasons, a legitimate security failure such as due to an attack or compromise of some sort, or as a result of misconfiguration or other error or omission on the part of an domain administrator. As domains transition to DNSSEC the most likely reason for a validation failure during and shortly after the transition is likely due to misconfiguration. Thus, domain administrators should be sure to read [RFC6781] in full. They should also pay special attention to Section 4.2, pertaining to key rollovers, which appears to be the cause of many validation failures.

In one example [DNSSEC-Validation-Failure-Analysis], a specific domain name failed to validate. An investigation revealed that the domain's administrators performed a Key Signing Key (KSK) rollover by (1) generating a new key and (2) signing the domain with the new key. However, they did not use a double-signing procedure for the KSK and a pre-publish procedure for the ZSK. Double-signing refers to signing a zone with two KSKs and then updating the parent zone with the new DS record so that both keys are valid at the same time. This meant that the domain name was signed with the new KSK, but it was not double-signed with the old KSK. So, the new key was used for signing the zone but the old key was not. As a result, the domain could not be trusted and returned an error when trying to reach the domain. Thus, the domain was in a situation where the DNSSEC chain of trust was broken because the Delegation Signer (DS) record pointed to the old KSK, which was no longer used for signing the zone. (A DS

record provides a link in the chain of trust for DNSSEC from the parent zone to the child zone - in this case between TLD and domain name.)

In a non-DNSSEC-related example, a domain administrator may add a new host with an A and AAAA resource record pointing the name to the IP addresses of new servers with a Time To Live (TTL) of two days. But they may turn down the old servers with a similar two day TTL before that TTL has expired. As a result, some number of users are likely to continue to attempt to connect to the old IP addresses that are no longer reachable. While a best practice is to reduce the TTL to a matter of seconds or minutes before such a shift, many domains continue to forget the impact that the TTL can have, or make countless other errors in their domain name, server, and network administration that negatively impacts domain name-based reachability.

3. Responsibility for Failures

An authoritative domain owner is solely and completely responsible for managing their domain name(s) and associated DNS resource records. This includes complete responsibility for the correctness of those resource records, the proper functioning and reachability of their authoritative DNS servers, and the correctness of DNS records linking their domain to a top-level domain (TLD) or other higher level domain. The domain owner is also responsible for selection of the authoritative domain administrator, operator, or service provider. Thus, even in cases where some error may be introduced by a third party, whether that is due to an authoritative server software vendor, software tools vendor, domain name registrar, Content Delivery Network (CDN), or other organization, these are all parties that the domain owner has selected and is responsible for managing successfully.

There are some cases where the domain administrator is different than the domain owner. In those cases, a domain owner has delegated operational responsibility to the domain administrator (and that domain administrator may further delegate some sub-domains and/or records to another party, such as a CDN). So no matter whether a domain owner is also the domain administrator or not, the domain owner and domain administrator are nevertheless operationally responsible for the proper configuration and operation of the domain name .

In the case of a domain name failing DNSSEC validation, even when this is due to a misconfiguration of the domain, that is the sole responsibility of the domain owner.

Any assistance or mitigation responses undertaken by other parties to mitigate the misconfiguration of a domain name by a domain owner and/or administrator, especially operators of DNS recursive resolvers, are optional and at the pleasure of those parties. This can be the use of a Negative Trust Anchor [RFC7646] and/or clearing the cache in particular DNS resolvers.

4. Comparison to Other DNS Misconfigurations

As noted in Section 3 domain administrators are ultimately responsible for managing and ensuring their DNS records are configured correctly. ISPs or other DNS recursive resolver operators cannot and should not correct misconfigured A, CNAME, MX, or other resource records of domains for which they are not authoritative. Expecting non-authoritative entities to protect domain owners and administrators from any misconfiguration of resource records is therefore unrealistic and unreasonable, does not scale well, and is strongly contrary to the delegated design of the DNS and could lead to extensive operational instability and/or variation.

5. Other Considerations

5.1. Security Considerations

Authoritative domain name owners and/or administrators, in the case of DNSSEC-related mistakes that cause validation failures to occur, should focus on correcting the immediate authoritative DNS issue and then improving their processes and tools in the future.

During the period of time that their domain cannot be resolved due to a DNSSEC-related mistake, they SHOULD NOT encourage end users to switch to non-validating resolvers [I-D.draft-livingood-dnsop-dont-switch-resolvers] .

5.2. Privacy Considerations

In the case of a DNSSEC validation failure, if an end user changes to a non-validating resolver they can subject themselves to increased security risks and threats against which DNSSEC may have provided protection. This can include threats to their privacy, such as by unwittingly visiting a phishing site and sharing sensitive data or other private information with a malicious party or some party other than that which was originally intended.

As a result, in order to protect their privacy, users SHOULD NOT switch to a non-validating resolver when a DNSSEC validation failure occurs [I-D.draft-livingood-dnsop-dont-switch-resolvers].

5.3. IANA Considerations

There are no IANA considerations in this document.

6. Acknowledgements

- William Brown

7. References

7.1. 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>>.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [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>>.
- [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>>.
- [RFC4398] Josefsson, S., "Storing Certificates in the Domain Name System (DNS)", RFC 4398, DOI 10.17487/RFC4398, March 2006, <<https://www.rfc-editor.org/info/rfc4398>>.
- [RFC4509] Hardaker, W., "Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records (RRs)", RFC 4509, DOI 10.17487/RFC4509, May 2006, <<https://www.rfc-editor.org/info/rfc4509>>.
- [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>>.

- [RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <<https://www.rfc-editor.org/info/rfc6781>>.
- [RFC7646] Ebersman, P., Kumari, W., Griffiths, C., Livingood, J., and R. Weber, "Definition and Use of DNSSEC Negative Trust Anchors", RFC 7646, DOI 10.17487/RFC7646, September 2015, <<https://www.rfc-editor.org/info/rfc7646>>.

7.2. Informative References

- [DNSSEC-Validation-Failure-Analysis]
Barnitz, J., Creighton, T., Ganster, C., Griffiths, C., and J. Livingood, "Analysis of DNSSEC Validation Failure - NASA.GOV", Comcast , January 2012, <http://www.dnssec.comcast.net/DNSSEC_Validation_Failure_NASAGOV_20120118_FINAL.pdf>.
- [I-D.livingood-dnsop-dont-switch-resolvers]
Livingood, J., "In Case of DNSSEC Validation Failures, Do Not Change Resolvers", draft-livingood-dnsop-dont-switch-resolvers-03 (work in progress), November 2015.

Appendix A. Document Change Log

- [RFC Editor: This section is to be removed before publication]
- Individual-00: First version published as an individual draft.
- Individual-01: Fixed nits identified by William Brown
- Individual-02: Updated prior to IETF-91
- WG-00: Renamed at request of DNSOP co-chairs
- WG-01: Updated doc to keep it from expiring
- WG-02: Removed RFC 2119 reference in XML
- WG-03 to 04: Refreshed draft and broadened to all auth issues, not just DNSSEC.
- WG-05: Refreshed to buy time for me to write a combined document

Appendix B. Open Issues

[RFC Editor: This section is to be removed before publication]

Fix I-D xref

Author's Address

Jason Livingood
Comcast

Email: jason_livingood@comcast.com

Domain Name System Operations
Internet-Draft
Intended status: Informational
Expires: February 14, 2020

J. Livingood
Comcast
August 13, 2019

In Case of DNSSEC Validation Failures, Do Not Change Resolvers
draft-livingood-dnsop-dont-switch-resolvers-05

Abstract

DNS Security Extensions (DNSSEC) validation by recursive DNS resolvers has been deployed at scale. However, domain signing tools and processes are not yet as mature and reliable as is the case for non-DNSSEC-related domain administration tools and processes. This sometimes results in DNSSEC validation failures, for which operators of validating resolvers are often blamed. When these failures do occur, end users should not change to a non-validating DNS resolver, as that would downgrade their security. They should instead wait until the authoritative domain operator updates their DNS records to resolve the error and that change propagates across the Internet's DNS resolvers, the timing of which may be dependent upon the Time To Live (TTL) settings in the old and/or erroneous DNS resource records.

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 February 14, 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
2. Reasons for DNSSEC Validation Failure	3
3. Misunderstanding DNSSEC Validation Failures	3
4. Comparison to Other DNS Misconfigurations	4
5. Switching to a Non-Validating Resolver is NOT Recommended . .	4
6. Other Considerations	5
6.1. Recommendations for Validating Resolver Operators	5
6.2. Security Considerations	5
6.3. Privacy Considerations	6
6.4. IANA Considerations	6
7. Acknowledgements	6
8. References	6
8.1. Normative References	6
8.2. URIs	7
Appendix A. Document Change Log	8
Appendix B. Open Issues	8
Author's Address	8

1. Introduction

The Domain Name System (DNS), DNS Security Extensions (DNSSEC), and related operational practices are defined extensively [RFC1034] [RFC1035] [RFC4033] [RFC4034] [RFC4035] [RFC4398] [RFC4509] [RFC6781] [RFC5155].

DNS Security Extensions (DNSSEC) validation by recursive DNS resolvers has been deployed at scale. However, domain signing tools and processes are not yet as mature and reliable as is the case for non-DNSSEC-related domain administration tools and processes. This sometimes results in DNSSEC validation failures, for which operators of validating resolvers are often blamed.

When these DNSSEC validation failures do occur, end users SHOULD NOT change to a non-validating DNS resolver, as that would downgrade their security. They should instead wait until the authoritative domain operator updates their DNS records to resolve the error and then for that change to propagate across the Internet's DNS

resolvers, the timing of which may be dependent upon the Time To Live (TTL) settings in the old and/or erroneous DNS resource records.

This document is necessary because it has become commonplace for reporters, technical users, and others to recommend that people change to non-validating resolvers when a DNSSEC validation failure occurs. This is NOT a recommended practice, it actively downgrades user security, and it reduces the incentives for authoritative domain operators to improve their DNSSEC-related domain administration tools and processes.

As a result, this document provides an authoritative reference point to recommend that users SHOULD NOT change DNS resolvers when DNSSEC validation failures occur. Such errors may be due to genuine security problems, which DNSSEC validation was designed to protect against. In the same way that a Transport Layer Security (TLS) [RFC8446] certificate failure should not be bypassed or ignored, so too that DNSSEC validation failures should not be bypassed or ignored.

2. Reasons for DNSSEC Validation Failure

A domain name can fail DNSSEC validation for two general reasons: an actual security failure such as due to an attack or compromise of some sort, or as a result of misconfiguration (mistake) on the part of a domain administrator. There is no way for an average end user to discern which of these issues has caused a DNSSEC-signed domain to fail validation, and so end users should therefore assume that it is due to an actual security problem as the most conservative and security-protective approach.

3. Misunderstanding DNSSEC Validation Failures

End users may incorrectly interpret the failure to reach a domain due to DNSSEC-related misconfiguration as their ISP or DNS resolver operator purposely blocking access to the domain, or as a performance-related failure on the part of that ISP or DNS resolver operator. In reality, these failures may be due to a security issue of which the end user is not aware. If a user ignores such a failure, or is instructed to ignore it, and switches to a non-validating resolver, they may be subject to the risk of malware exposure, phishing attack, and so on. The root cause of a DNSSEC validation failure lies not with a recursive DNS operator but with the authoritative domain name owner or administrator [I-D.draft-livingood-dnsop-auth-dnssec-mistakes] .

4. Comparison to Other DNS Misconfigurations

Authoritative DNS-related mistakes and errors typically affect the entire Internet, and all DNS recursive resolver operators equally. So for example, if an A record is incorrect, an end user would get the incorrect record in a DNS response no matter what resolver they used.

In contrast to this, DNSSEC-related mistakes, errors, or other validation security failures would only affect end users of those validating resolvers. That being said, different validating resolver operators may configure their servers slightly differently, have different server software, or have different server configurations, which can result in slightly different resolver validation behavior. It can also be the case that one resolver has cached a DNS resource record according to the TTL set by the authoritative domain administrator, while another resolver does not have that record cached (generally due to the timing of prior user queries for that name), which can also cause two resolvers to differ. Another reason for resolution variance may be that the authoritative DNS servers are responding differently to various DNS resolvers, perhaps to geographic differences, the nature of any delegations to Content Delivery Networks (CDNs), a regionally-focused Denial of Service (DoS) attack against an authoritative server, or a wide range of other potential reasons.

5. Switching to a Non-Validating Resolver is NOT Recommended

As noted in Section 3 some end users may not understand why a domain fails to validate on one network but not another (or with one DNS resolver but not another) Section 4. As a result, they may consider or someone may recommend to them switching to an alternative, non-validating resolver themselves. But if a domain fails DNSSEC validation and is inaccessible, this could very well be due to a security-related issue. Changing to a non-validating resolver is a critical security downgrade and is NOT advised.

DNSSEC validation failures may be due to genuine security problems, which DNSSEC validation was designed to protect against. In the same way that a Transport Layer Security (TLS) [RFC8446] certificate failure should not be bypassed or ignored, so too that DNSSEC validation failures should not be bypassed or ignored.

As a recommended best practice: In order to be as safe and secure as possible, end users SHOULD NOT change to DNS resolvers that do not perform DNSSEC validation as a workaround when DNSSEC validation failures occur.

Even if a website in a domain seems to look "normal" and valid, according to the DNSSEC protocol, that domain is not secure. Domains that fail DNSSEC validation may fail due to an actual security incident or compromise, and may be in control of hackers or there could be other significant security issues with the domain. Thus, switching to a non-validating resolver to restore access to a domain that fails DNSSEC validation is NOT recommended and is potentially harmful to end user security.

6. Other Considerations

6.1. Recommendations for Validating Resolver Operators

Since it is not recommended that end users change to non-validating resolvers, operators of validating resolvers may wish to consider what tools they might make available to their end users to assist in these cases. For example, there may be a DNS looking glass that enables someone to use a web page or other tool to remotely (including from a different network) check DNS resolution on the operator's servers, as well as possibly another operator's servers. Such a web page or tool may also provide a link to independent third party sites or tools that can confirm whether or not a DNSSEC-related error is present, of which several exist today (e.g. DNSViz [1], Verisign DNSSEC Debugger [2]). Finally, the operator may also wish to consider a web page form or other tool to enable end users to report possible DNS resolution issues.

Resolver operators may also find it helpful to selectively use a Negative Trust Anchor [RFC7646] to temporarily mitigate validation failures that are absolutely confirmed to be due to authoritative domain name administration error by that administrator. In addition, in select cases such as a very high traffic domain name, once an administrative DNS error or problem has been fixed a resolver may consider clearing the cache of their recursive resolvers in order to pickup the authoritative change immediately (rather than waiting until the TTL on a cached record expires).

6.2. Security Considerations

The use of a non-validating DNS recursive resolver is comparatively less secure than using a validating resolver, since one implements DNS Security Extensions (DNSSEC) and one does not.

In the case of a DNSSEC validation failure, if an end user changes to a non-validating resolver they can subject themselves to increased security risks and threats against which DNSSEC may have provided protection.

As a result, in order to protect their security, users SHOULD NOT switch to a non-validating resolver when a DNSSEC validation failure occurs.

6.3. Privacy Considerations

In the case of a DNSSEC validation failure, if an end user changes to a non-validating resolver they can subject themselves to increased security risks and threats against which DNSSEC may have provided protection. This can include threats to their privacy, such as by unwittingly visiting a phishing site and sharing sensitive data or other private information with a malicious party or some party other than that which was originally intended.

As a result, in order to protect their privacy, users SHOULD NOT switch to a non-validating resolver when a DNSSEC validation failure occurs.

6.4. IANA Considerations

There are no IANA considerations in this document.

7. Acknowledgements

- William Brown
- Peter Koch

8. References

8.1. 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>>.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.

- [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>>.
- [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>>.
- [RFC4398] Josefsson, S., "Storing Certificates in the Domain Name System (DNS)", RFC 4398, DOI 10.17487/RFC4398, March 2006, <<https://www.rfc-editor.org/info/rfc4398>>.
- [RFC4509] Hardaker, W., "Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records (RRs)", RFC 4509, DOI 10.17487/RFC4509, May 2006, <<https://www.rfc-editor.org/info/rfc4509>>.
- [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>>.
- [RFC5914] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Format", RFC 5914, DOI 10.17487/RFC5914, June 2010, <<https://www.rfc-editor.org/info/rfc5914>>.
- [RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <<https://www.rfc-editor.org/info/rfc6781>>.
- [RFC7646] Ebersman, P., Kumari, W., Griffiths, C., Livingood, J., and R. Weber, "Definition and Use of DNSSEC Negative Trust Anchors", RFC 7646, DOI 10.17487/RFC7646, September 2015, <<https://www.rfc-editor.org/info/rfc7646>>.
- [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>>.

8.2. URIs

[1] <http://dnsviz.net/>

[2] <http://dnssec-debugger.verisignlabs.com/>

Appendix A. Document Change Log

[RFC Editor: This section is to be removed before publication]

Individual-00: First version published as an individual draft.

Individual-01: Fixed nits identified by William Brown

Individual-02: Updated prior to IETF-91

WG-00: Renamed at request of DNSOP co-chairs

WG-01: Updated doc to keep it from expiring

WG-02: Addressed some feedback from Peter Koch on RFC 2119 text, changed from BCP to Informational since this is more a recommended practice, added a section with recommendations for operators.

WG-03 to 04: Refreshed document

WG-05: Refreshed to buy time for me to write a combined document

Appendix B. Open Issues

[RFC Editor: This section is to be removed before publication]

Fix I-D xref

Author's Address

Jason Livingood
Comcast

Email: jason_livingood@comcast.com

DNSOP Working Group
Internet-Draft
Intended status: Informational
Expires: 8 July 2022

G.C.M. Moura
SIDN Labs/TU Delft
W. Hardaker
J. Heidemann
USC/Information Sciences Institute
M. Davids
SIDN Labs
4 January 2022

Considerations for Large Authoritative DNS Servers Operators
draft-moura-dnsop-authoritative-recommendations-11

Abstract

Recent research work has explored the deployment characteristics and configuration of the Domain Name System (DNS). This document summarizes the conclusions from these research efforts and offers specific, tangible considerations or advice to authoritative DNS server operators. Authoritative server operators may wish to follow these considerations to improve their DNS services.

It is possible that the results presented in this document could be applicable in a wider context than just the DNS protocol, as some of the results may generically apply to any stateless/short-duration, anycasted service.

This document is not an IETF consensus document: it is published for informational purposes.

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 8 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	3
2. Background	3
3. Considerations	5
3.1. C1: Deploy anycast in every authoritative server to enhance distribution and latency	5
3.1.1. Research background	5
3.1.2. Resulting considerations	6
3.2. C2: Optimizing routing is more important than location count and diversity	7
3.2.1. Research background	7
3.2.2. Resulting considerations	8
3.3. C3: Collecting anycast catchment maps to improve design	8
3.3.1. Research background	8
3.3.2. Resulting considerations	9
3.4. C4: When under stress, employ two strategies	9
3.4.1. Research background	10
3.4.2. Resulting considerations	11
3.5. C5: Consider longer time-to-live values whenever possible	11
3.5.1. Research background	11
3.5.2. Resulting considerations	13
3.6. C6: Consider the TTL differences between parents and children	14
3.6.1. Research background	14
3.6.2. Resulting considerations	15
4. Security considerations	15
5. Privacy Considerations	15
6. IANA considerations	15
7. Acknowledgements	15
8. References	16
8.1. Normative References	16

8.2. Informative References	17
Authors' Addresses	20

1. Introduction

This document summarizes recent research work that explored the deployed DNS configurations and offers derived, specific tangible advice to DNS authoritative server operators (DNS operators hereafter). The considerations (C1--C5) presented in this document are backed by peer-reviewed research works, which used wide-scale Internet measurements to draw their conclusions. This document summarizes the research results and describes the resulting key engineering options. In each section, it points readers to the pertinent publications where additional details are presented.

These considerations are designed for operators of "large" authoritative DNS servers. In this context, "large" authoritative servers refers to those with a significant global user population, like top-level domain (TLD) operators, run by either a single or multiple operators. Typically these networks are deployed on wide anycast networks [RFC1546][AnyBest]. These considerations may not be appropriate for smaller domains, such as those used by an organization with users in one unicast network, or in one city or region, where operational goals such as uniform, global low latency are less required.

It is possible that the results presented in this document could be applicable in a wider context than just the DNS protocol, as some of the results may generically apply to any stateless/short-duration, anycasted service. Because the conclusions of the reviewed studies don't measure smaller networks, the wording in this document concentrates solely on disusing large-scale DNS authoritative services only.

This document is not an IETF consensus document: it is published for informational purposes.

2. Background

The DNS has main two types of DNS servers: authoritative servers and recursive resolvers, shown by a representational deployment model in Figure 1. An authoritative server (shown as AT1--AT4 in Figure 1) knows the content of a DNS zone, and is responsible for answering queries about that zone. It runs using local (possibly automatically updated) copies of the zone and does not need to query other servers [RFC2181] in order to answer requests. A recursive resolver (Rel--Re3) is a server that iteratively queries authoritative and other servers to answer queries received from client requests [RFC1034]. A

client typically employs a software library called a stub resolver (stub in Figure 1) to issue its query to the upstream recursive resolvers [RFC1034].

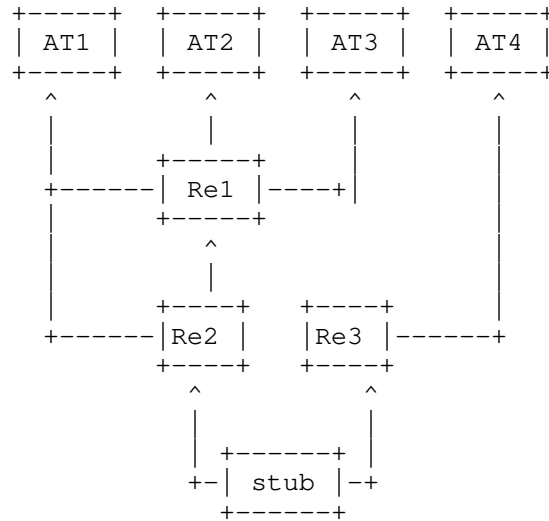


Figure 1: Relationship between recursive resolvers (Re) and authoritative name servers (ATn)

DNS queries issued by a client contribute to a user's perceived latency and affect user experience [Singla2014] depending on how long it takes for responses to be returned. The DNS system has been subject to repeated Denial of Service (DoS) attacks (for example, in November 2015 [Moural6b]) in order to specifically degrade user experience.

To reduce latency and improve resiliency against DoS attacks, the DNS uses several types of service replication. Replication at the authoritative server level can be achieved with (i) the deployment of multiple servers for the same zone [RFC1035] (AT1---AT4 in Figure 1), (ii) the use of IP anycast [RFC1546][RFC4786][RFC7094] that allows the same IP address to be announced from multiple locations (each of referred to as an "anycast instance" [RFC8499]) and (iii) the use of load balancers to support multiple servers inside a single (potentially anycasted) instance. As a consequence, there are many possible ways an authoritative DNS provider can engineer its production authoritative server network, with multiple viable choices and no necessarily single optimal design.

3. Considerations

In the next sections we cover the specific consideration (C1--C6) for conclusions drawn within the academic papers about large authoritative DNS server operators. These considerations are conclusions reached from academic works that authoritative server operators may wish to consider in order to improve their DNS service. Each consideration offers different improvements that may impact service latency, routing, anycast deployment, and defensive strategies for example.

3.1. C1: Deploy anycast in every authoritative server to enhance distribution and latency

3.1.1. Research background

Authoritative DNS server operators announce their service using NS records[RFC1034]. Different authoritative servers for a given zone should return the same content; typically they stay synchronized using DNS zone transfers (AXFR[RFC5936] and IXFR[RFC1995]), coordinating the zone data they all return to their clients.

As discussed above, the DNS heavily relies upon replication to support high reliability, ensure capacity and to reduce latency [Moural6b]. DNS has two complementary mechanisms for service replication: nameserver replication (multiple NS records) and anycast (multiple physical locations). Nameserver replication is strongly recommended for all zones (multiple NS records), and IP anycast is used by many larger zones such as the DNS Root[AnyFRoot], most top-level domains[Moural6b] and many large commercial enterprises, governments and other organizations.

Most DNS operators strive to reduce service latency for users, which is greatly affected by both of these replication techniques. However, because operators only have control over their authoritative servers, and not over the client's recursive resolvers, it is difficult to ensure that recursives will be served by the closest authoritative server. Server selection is ultimately up to the recursive resolver's software implementation, and different vendors and even different releases employ different criteria to chose the authoritative servers with which to communicate.

Understanding how recursive resolvers choose authoritative servers is a key step in improving the effectiveness of authoritative server deployments. To measure and evaluate server deployments, [Mueller17b] deployed seven unicast authoritative name servers in different global locations and then queried them from more than 9000 RIPE authoritative server operators and their respective recursive resolvers.

[Mueller17b] found that recursive resolvers in the wild query all available authoritative servers, regardless of the observed latency. But the distribution of queries tends to be skewed towards authoritatives with lower latency: the lower the latency between a recursive resolver and an authoritative server, the more often the recursive will send queries to that server. These results were obtained by aggregating results from all of the vantage points and were not specific to any specific vendor or version.

The authors believe this behavior is a consequence of combining the two main criteria employed by resolvers when selecting authoritative servers: resolvers regularly check all listed authoritative servers in an NS set to determine which is closer (the least latent) and when one isn't available selects one of the alternatives.

3.1.2. Resulting considerations

For an authoritative DNS operator, this result means that the latency of all authoritative servers (NS records) matter, so they all must be similarly capable -- all available authoritatives will be queried by most recursive resolvers. Unicast services, unfortunately, cannot deliver good latency worldwide (a unicast authoritative server in Europe will always have high latency to resolvers in California and Australia, for example, given its geographical distance).

[Mueller17b] recommends that DNS operators deploy equally strong IP anycast instances for every authoritative server (i.e., for each NS record). Each large authoritative DNS server provider should phase out their usage of unicast and deploy a well engineered number of anycast instances with good peering strategies so they can provide good latency to their global clients.

As a case study, the ".nl" TLD zone was originally served on seven authoritative servers with a mixed unicast/anycast setup. In early 2018, .nl moved to a setup with 4 anycast authoritative servers.

[Mueller17b]'s contribution to DNS service engineering shows that because unicast cannot deliver good latency worldwide, anycast needs to be used to provide a low latency service worldwide.

3.2. C2: Optimizing routing is more important than location count and diversity

3.2.1. Research background

When selecting an anycast DNS provider or setting up an anycast service, choosing the best number of anycast instances[RFC4786][RFC7094] to deploy is a challenging problem. Selecting where and how many global locations to announce from using BGP is tricky. Intuitively, one could naively think that the more instances the better and simply "more" will always lead to shorter response times.

This is not necessarily true, however. In fact, [Schmidt17a] found that proper route engineering can matter more than the total number of locations. They analyzed the relationship between the number of anycast instances and service performance (measuring latency of the round-trip time (RTT)), measuring the overall performance of four DNS Root servers. The Root DNS servers are implemented by 12 separate organizations serving the DNS root zone at 13 different IPv4/IPv6 address pairs.

The results documented in [Schmidt17a] measured the performance of the {c,f,k,l}.root-servers.net (hereafter, "C", "F", "K" and "L") servers from more than 7.9k RIPE Atlas probes. RIPE Atlas is a Internet measurement platform with more than 12000 global vantage points called "Atlas Probes" -- it is used regularly by both researchers and operators [RipeAtlas15a] [RipeAtlas19a].

[Schmidt17a] found that the C server, a smaller anycast deployment consisting of only 8 instances, provided very similar overall performance in comparison to the much larger deployments of K and L, with 33 and 144 instances respectively. The median RTT for C, K and L root server were all between 30-32ms.

Because RIPE Atlas is known to have better coverage in Europe than other regions, the authors specifically analyzed the results per region and per country (Figure 5 in [Schmidt17a]), and show that known Atlas bias toward Europe does not change the conclusion that properly selected anycast locations is more important to latency than the number of sites.

3.2.2. Resulting considerations

The important conclusion of [Schmidt17a] is that when engineering anycast services for performance, factors other than just the number of instances (such as local routing connectivity) must be considered. Specifically, optimizing routing policies is more important than simply adding new instances. They showed that 12 instances can provide reasonable latency, assuming they are globally distributed and have good local interconnectivity. However, additional instances can still be useful for other reasons, such as when handling Denial-of-service (DoS) attacks [Moural6b].

3.3. C3: Collecting anycast catchment maps to improve design

3.3.1. Research background

An anycast DNS service may be deployed from anywhere from several locations to hundreds of locations (for example, `l.root-servers.net` has over 150 anycast instances at the time this was written). Anycast leverages Internet routing to distribute incoming queries to a service's hop-nearest distributed anycast locations. However, usually queries are not evenly distributed across all anycast locations, as found in the case of L-Root [IcannHedge18].

Adding locations to or removing locations from a deployed anycast network changes the load distribution across all of its locations. When a new location is announced by BGP, locations may receive more or less traffic than it was engineered for, leading to suboptimal service performance or even stressing some locations while leaving others underutilized. Operators constantly face this scenario that when expanding an anycast service. Operators cannot easily directly estimate future query distributions based on proposed anycast network engineering decisions.

To address this need and estimate the query loads based on changing, in particular expanding, anycast service changes [Vries17b] developed a new technique enabling operators to carry out active measurements, using an open-source tool called Verfploeter (available at [VerfSrc]). The results allow the creation of detailed anycast maps and catchment estimates. By running verfploeter combined with a published IPv4 "hit list", DNS can precisely calculate which remote prefixes will be matched to each anycast instance in a network. At the moment of this writing, Verfploeter still does not support IPv6 as the IPv4 hit lists used are generated via frequent large scale ICMP echo scans, which is not possible using IPv6.

As proof of concept, [Vries17b] documents how it `verfploeter` was used to predict both the catchment and query load distribution for a new anycast instance deployed for `b.root-servers.net`. Using two anycast test instances in Miami (MIA) and Los Angeles (LAX), an ICMP echo query was sent from an IP anycast addresses to each IPv4 /24 network routing block on the Internet.

The ICMP echo responses were recorded at both sites and analyzed and overlaid onto a graphical world map, resulting in an Internet scale catchment map. To calculate expected load once the production network was enabled, the quantity of traffic received by `b.root-servers.net`'s single site at LAX was recorded based on a single day's traffic (2017-04-12, DITL datasets [Ditl17]). [Vries17b] predicted that 81.6% of the traffic load would remain at the LAX site. This estimate by `verfploeter` turned out to be very accurate; the actual measured traffic volume when production service at MIA was enabled was 81.4%.

`Verfploeter` can also be used to estimate traffic shifts based on other BGP route engineering techniques (for example, AS path prepending or BGP community use) in advance of operational deployment. [Vries17b] studied this using prepending with 1-3 hops at each instance and compared the results against real operational changes to validate the techniques accuracy.

3.3.2. Resulting considerations

An important operational takeaway [Vries17b] provides is how DNS operators can make informed engineering choices when changing DNS anycast network deployments by using `Verfploeter` in advance. Operators can identify sub-optimal routing situations in advance with significantly better coverage than using other active measurement platforms such as RIPE Atlas. To date, `Verfploeter` has been deployed on a operational testbed (Anycast testbed) [AnyTest], on a large unnamed operator and is run daily at `b.root-servers.net` [Vries17b].

Operators should use active measurement techniques like `Verfploeter` in advance of potential anycast network changes to accurately measure the benefits and potential issues ahead of time.

3.4. C4: When under stress, employ two strategies

3.4.1. Research background

DDoS attacks are becoming bigger, cheaper, and more frequent [Moural6b]. The most powerful recorded DDoS attack against DNS servers to date reached 1.2 Tbps by using IoT devices [Perlroth16]. How should a DNS operator engineer its anycast authoritative DNS server react to such a DDoS attack? [Moural6b] investigates this question using empirical observations grounded with theoretical option evaluations.

An authoritative DNS server deployed using anycast will have many server instances distributed over many networks. Ultimately, the relationship between the DNS provider's network and a client's ISP will determine which anycast instance will answer queries for a given client, given that BGP is the protocol that maps clients to specific anycast instances by using routing information [RF:KDar02]. As a consequence, when an anycast authoritative server is under attack, the load that each anycast instance receives is likely to be unevenly distributed (a function of the source of the attacks), thus some instances may be more overloaded than others which is what was observed analyzing the Root DNS events of Nov. 2015 [Moural6b]. Given the fact that different instances may have different capacity (bandwidth, CPU, etc.), making a decision about how to react to stress becomes even more difficult.

In practice, an anycast instance is overloaded with incoming traffic, operators have two options:

- * They can withdraw its routes, pre-prepend its AS route to some or all of its neighbors, perform other traffic shifting tricks (such as reducing route announcement propagation using BGP communities[RFC1997]), or by communicating with its upstream network providers to apply filtering (potentially using FlowSpec [RFC8955] or DOTS protocol ([RFC8811], [RFC8782], [RFC8783])). These techniques shift both legitimate and attack traffic to other anycast instances (with hopefully greater capacity) or to block traffic entirely.
- * Alternatively, operators can become a degraded absorber by continuing to operate, knowing dropping incoming legitimate requests due to queue overflow. However, this approach will also absorb attack traffic directed toward its catchment, hopefully protecting the other anycast instances.

[Moural6b] saw both of these behaviors deployed in practice by studying instance reachability and route-trip time (RTTs) in the DNS root events. When withdraw strategies were deployed, the stress of increased query loads were displaced from one instance to multiple

other sites. In other observed events, one site was left to absorb the brunt of an attack leaving the other sites to remain relatively less affected.

3.4.2. Resulting considerations

Operators should consider having both a anycast site withdraw strategy and a absorption strategy ready to be used before a network overload occurs. Operators should be able to deploy one or both of these strategies rapidly. Ideally, these should be encoded into operating playbooks with defined site measurement guidelines for which strategy to employ based on measured data from past events.

[Moural6b] speculates that careful, explicit, and automated management policies may provide stronger defenses to overload events. DNS operators should be ready to employ both traditional filtering approaches and other routing load balancing techniques (withdraw/prepend/communities or isolate instances), where the best choice depends on the specifics of the attack.

Note that this consideration refers to the operation of just one anycast service point, i.e., just one anycasted IP address block covering one NS record. However, DNS zones with multiple authoritative anycast servers may also expect loads to shift from one anycasted server to another, as resolvers switch from one authoritative service point to another when attempting to resolve a name [Mueller17b].

3.5. C5: Consider longer time-to-live values whenever possible

3.5.1. Research background

Caching is the cornerstone of good DNS performance and reliability. A 50 ms response to a new DNS query may be considered fast, but a less than 1 ms response to a cached entry is far faster. [Moural8b] showed that caching also protects users from short outages and even significant DDoS attacks.

DNS record TTLs (time-to-live values) [RFC1034][RFC1035] directly control cache durations and affect latency, resilience, and the role of DNS in CDN server selection. Some early work modeled caches as a function of their TTLs [Jung03a], and recent work has examined their interaction with DNS[Moural8b], but until [Moural9b] no research provided considerations about the benefits of various TTL value choices. To study this, Moura et. al. [Moural9b] carried out a measurement study investigating TTL choices and their impact on user experiences in the wild. They performed this study independent of specific resolvers (and their caching architectures), vendors, or setups.

First, they identified several reasons why operators and zone-owners may want to choose longer or shorter TTLs:

- * As discussed, longer TTLs lead to a longer cache life, resulting in faster responses. [Moural9b] measured this in the wild and showed that by increasing the TTL for .uy TLD from 5 minutes (300s) to 1 day (86400s) the latency measured from 15k Atlas vantage points changed significantly: the median RTT decreased from 28.7ms to 8ms, and the 75%ile decreased from 183ms to 21ms.
- * Longer caching times also results in lower DNS traffic: authoritative servers will experience less traffic with extended TTLs, as repeated queries are answered by resolver caches.
- * Consequently, longer caching results in a lower overall cost if DNS is metered: some DNS-As-A-Service providers charge a per query (metered) cost (often in addition to a fixed monthly cost).
- * Longer caching is more robust to DDoS attacks on DNS infrastructure. [Moural8b] also measured and show that DNS caching can greatly reduce the effects of a DDoS on DNS, provided that caches last longer than the attack.
- * However, shorter caching supports deployments that may require rapid operational changes: An easy way to transition from an old server to a new one is to simply change the DNS records. Since there is no method to remotely remove cached DNS records, the TTL duration represents a necessary transition delay to fully shift from one server to another. Thus, low TTLs allow for more rapid transitions. However, when deployments are planned in advance (that is, longer than the TTL), it is possible to lower the TTLs just-before a major operational change and raise them again afterward.

- * Shorter caching can also help with a DNS-based response to DDoS attacks. Specifically, some DDoS-scrubbing services use the DNS to redirect traffic during an attack. Since DDoS attacks arrive unannounced, DNS-based traffic redirection requires the TTL be kept quite low at all times to allow operators to suddenly have their zone served by a DDoS-scrubbing service.
- * Shorter caching helps DNS-based load balancing. Many large services are known to rotate traffic among their servers using DNS-based load balancing. Each arriving DNS request provides an opportunity to adjust service load by rotating IP address records (A and AAAA) to the lowest unused server. Shorter TTLs may be desired in these architectures to react more quickly to traffic dynamics. Many recursive resolvers, however, have minimum caching times of tens of seconds, placing a limit on this form of agility.

3.5.2. Resulting considerations

Given these considerations, the proper choice for a TTL depends in part on multiple external factors -- no single recommendation is appropriate for all scenarios. Organizations must weigh these trade-offs and find a good balance for their situation. Still, some guidelines can be reached when choosing TTLs:

- * For general DNS zone owners, [Moural9b] recommends a longer TTL of at least one hour, and ideally 8, 12, or 24 hours. Assuming planned maintenance can be scheduled at least a day in advance, long TTLs have little cost and may, even, literally provide a cost savings.
- * For registry operators: TLD and other public registration operators (for example most ccTLDs and .com, .net, .org) that host many delegations (NS records, DS records and "glue" records), [Moural9b] demonstrates that most resolvers will use the TTL values provided by the child delegations while the others some will choose the TTL provided by the parent's copy of the record. As such, [Moural9b] recommends longer TTLs (at least an hour or more) for registry operators as well for child NS and other records.
- * Users of DNS-based load balancing or DDoS-prevention services may require shorter TTLs: TTLs may even need to be as short as 5 minutes, although 15 minutes may provide sufficient agility for many operators. There is always a tussle between shorter TTLs providing more agility against all the benefits listed above for using longer TTLs.

- * Use of A/AAAA and NS records: The TTLs for A/AAAA records should be shorter to or equal to the TTL for the corresponding NS records for in-bailiwick authoritative DNS servers, since [Moural9b] finds that once an NS record expires, their associated A/AAAA will also be re-queried when glue is required to be sent by the parents. For out-of-bailiwick servers, A, AAAA and NS records are usually all cached independently, so different TTLs can be used effectively if desired. In either case, short A and AAAA records may still be desired if DDoS-mitigation services are required.

3.6. C6: Consider the TTL differences between parents and children

3.6.1. Research background

Multiple record types exist or are related between the parent of a zone and the child. At a minimum, NS records are supposed to be identical in the parent (but often are not) as or corresponding IP address in "glue" A/AAAA records that must exist for in-bailiwick authoritative servers. Additionally, if DNSSEC ([RFC4033] [RFC4034] [RFC4035] [RFC4509]) is deployed for a zone the parent's DS record must cryptographically refer to a child's DNSKEY record.

Because some information exists in both the parent and a child, it is possible for the TTL values to differ between the parent's copy and the child's. [Moural9b] examines resolver behaviors when these values differ in the wild, as they frequently do -- often parent zones have defacto TTL values that a child has no control over. For example, NS records for TLDs in the root zone are all set to 2 days (48 hours), but some TLD's have lower values within their published records (the TTLs for .cl's NS records from their authoritative servers is 1 hour). [Moural9b] also examines the differences in the TTLs between the NS records and the corresponding A/AAAA records for the addresses of a nameserver. RIPE Atlas nodes are used to determine what resolvers in the wild do with different information, and whether the parent's TTL is used for cache life-times ("parent-centric") or the child's is used ("child-centric").

[Moural9b] finds that roughly 90% of resolvers follow the child's view of the TTL, while 10% appear parent-centric. It additionally finds that resolvers behave differently for cache lifetimes for in-bailiwick vs out-of-bailiwick NS/A/AAAA TTL combinations. Specifically, when NS TTLs are shorter than the corresponding address records, most resolvers will re-query for A/AAAA records for in-bailiwick resolvers and switch to new address records even if the cache indicates the original A/AAAA records could be kept longer. On the other hand, the inverse is true for out-of-bailiwick resolvers: If the NS record expires first resolvers will honor the original cache time of the nameserver's address.

3.6.2. Resulting considerations

The important conclusion from this study is that operators cannot depend on their published TTL values alone -- the parent's values are also used for timing cache entries in the wild. Operators that are planning on infrastructure changes should assume that older infrastructure must be left on and operational for at least the maximum of both the parent and child's TTLs.

4. Security considerations

This document discusses applying measured research results to operational deployments. Most of the considerations affect mostly operational practice, though a few do have security related impacts.

Specifically, C4 discusses a couple of strategies to employ when a service is under stress from DDoS attacks and offers operators additional guidance when handling excess traffic.

Similarly, C5 identifies the trade-offs with respect to the operational and security benefits of using longer time-to-live values.

5. Privacy Considerations

This document does not add any practical new privacy issues, aside from possible benefits in deploying longer TTLs as suggested in C5. Longer TTLs may help preserve a user's privacy by reducing the number of requests that get transmitted in both the client-to-resolver and resolver-to-authoritative cases.

6. IANA considerations

This document has no IANA actions.

7. Acknowledgements

This document is a summary of the main considerations of six research works performed by the authors and others. This document would not have been possible without the hard work of these authors and co-authors:

- * Ricardo de O. Schmidt
- * Wouter B de Vries
- * Moritz Mueller

- * Lan Wei
- * Cristian Hesselman
- * Jan Harm Kuipers
- * Pieter-Tjerk de Boer
- * Aiko Pras

We would like also to thank the reviewers of this draft that offered valuable suggestions: Duane Wessels, Joe Abley, Toema Gavrichenkov, John Levine, Michael StJohns, Kristof Tuyteleers, Stefan Ubbink, Klaus Darilion and Samir Jafferli, and comments provided at the IETF DNSOP session (IETF104).

Besides those, we would like thank those acknowledged in the papers this document summarizes for helping produce the results: RIPE NCC and DNS OARC for their tools and datasets used in this research, as well as the funding agencies sponsoring the individual research works.

8. References

8.1. 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>>.
- [RFC1546] Partridge, C., Mendez, T., and W. Milliken, "Host Anycasting Service", RFC 1546, DOI 10.17487/RFC1546, November 1993, <<https://www.rfc-editor.org/info/rfc1546>>.
- [RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, DOI 10.17487/RFC1995, August 1996, <<https://www.rfc-editor.org/info/rfc1995>>.
- [RFC1997] Chandra, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, DOI 10.17487/RFC1997, August 1996, <<https://www.rfc-editor.org/info/rfc1997>>.

- [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>>.
- [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <<https://www.rfc-editor.org/info/rfc4786>>.
- [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>>.
- [RFC7094] McPherson, D., Oran, D., Thaler, D., and E. Osterweil, "Architectural Considerations of IP Anycast", RFC 7094, DOI 10.17487/RFC7094, January 2014, <<https://www.rfc-editor.org/info/rfc7094>>.
- [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>>.
- [RFC8782] Reddy.K, T., Ed., Boucadair, M., Ed., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", RFC 8782, DOI 10.17487/RFC8782, May 2020, <<https://www.rfc-editor.org/info/rfc8782>>.
- [RFC8783] Boucadair, M., Ed. and T. Reddy.K, Ed., "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", RFC 8783, DOI 10.17487/RFC8783, May 2020, <<https://www.rfc-editor.org/info/rfc8783>>.
- [RFC8955] Loibl, C., Hares, S., Raszuk, R., McPherson, D., and M. Bacher, "Dissemination of Flow Specification Rules", RFC 8955, DOI 10.17487/RFC8955, December 2020, <<https://www.rfc-editor.org/info/rfc8955>>.

8.2. Informative References

- [AnyBest] Woodcock, B., "Best Practices in DNS Service-Provision Architecture", March 2016, <<https://meetings.icann.org/en/marrakech55/schedule/mon-tech/presentation-dns-service-provision-07mar16-en.pdf>>.
- [AnyFRoot] Woolf, S., "Anycasting f.root-servers.net", January 2003, <<https://archive.nanog.org/meetings/nanog27/presentations/suzanne.pdf>>.

- [AnyTest] Schmidt, R.d.O., "Anycast Testbed", December 2018, <<http://www.anycast-testbed.com/>>.
- [Ditl17] OARC, D., "2017 DITL data", October 2018, <<https://www.dns-oarc.net/oarc/data/ditl/2017>>.
- [IcannHedge18] ICANN, ., "DNS-STATS - Hedgehog 2.4.1", October 2018, <<http://stats.dns.icann.org/hedgehog/>>.
- [Jung03a] Jung, J., Berger, A.W., and H. Balakrishnan, "Modeling TTL-based Internet caches", ACM 2003 IEEE INFOCOM, DOI 10.1109/INFCOM.2003.1208693, July 2003, <http://www.ieee-infocom.org/2003/papers/11_01.PDF>.
- [Moural6b] Moura, G.C.M., Schmidt, R.d.O., Heidemann, J., Mueller, M., Wei, L., and C. Hesselman, "Anycast vs DDoS Evaluating the November 2015 Root DNS Events.", ACM 2016 Internet Measurement Conference, DOI /10.1145/2987443.2987446, 14 October 2016, <<https://www.isi.edu/~johnh/PAPERS/Moural6b.pdf>>.
- [Moural8b] Moura, G.C.M., Heidemann, J., Mueller, M., Schmidt, R.d.O., and M. Davids, "When the Dike Breaks: Dissecting DNS Defenses During DDos", ACM 2018 Internet Measurement Conference, DOI 10.1145/3278532.3278534, 31 October 2018, <<https://www.isi.edu/~johnh/PAPERS/Moural8b.pdf>>.
- [Moural9b] Moura, G., Hardaker, W., Heidemann, J., and R.d.O. Schmidt, "Cache Me If You Can: Effects of DNS Time-to-Live", ACM 2019 Internet Measurement Conference, DOI 10.1145/3355369.3355568, n.d., <<https://www.isi.edu/~hardaker/papers/2019-10-cache-me-ttls.pdf>>.
- [Mueller17b] Mueller, M., Moura, G.C.M., Schmidt, R.d.O., and J. Heidemann, "Recursives in the Wild- Engineering Authoritative DNS Servers.", ACM 2017 Internet Measurement Conference, DOI 10.1145/3131365.3131366, October 2017, <<https://www.isi.edu/%7ejohnh/PAPERS/Mueller17b.pdf>>.
- [Perlroth16] Perlroth, N., "Hackers Used New Weapons to Disrupt Major Websites Across U.S.", October 2016, <<https://www.nytimes.com/2016/10/22/business/internet-problems-attack.html>>.

- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [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>>.
- [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>>.
- [RFC4509] Hardaker, W., "Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records (RRs)", RFC 4509, DOI 10.17487/RFC4509, May 2006, <<https://www.rfc-editor.org/info/rfc4509>>.
- [RFC8811] Mortensen, A., Ed., Reddy, K., T., Ed., Andreasen, F., Teague, N., and R. Compton, "DDoS Open Threat Signaling (DOTS) Architecture", RFC 8811, DOI 10.17487/RFC8811, August 2020, <<https://www.rfc-editor.org/info/rfc8811>>.
- [RipeAtlas15a] Staff, R.N., "RIPE Atlas A Global Internet Measurement Network", September 2015, <<http://ipj.dreamhosters.com/wp-content/uploads/issues/2015/ipjl8-3.pdf>>.
- [RipeAtlas19a] NCC, R., "Ripe Atlas - RIPE Network Coordination Centre", September 2019, <<https://atlas.ripe.net/>>.
- [Schmidt17a] Schmidt, R.d.O., Heidemann, J., and J.H. Kuipers, "Anycast Latency - How Many Sites Are Enough. In Proceedings of the Passive and Active Measurement Workshop", PAM Passive and Active Measurement Conference, March 2017, <<https://www.isi.edu/%7ejohnh/PAPERS/Schmidt17a.pdf>>.

[Singla2014]

Singla, A., Chandrasekaran, B., Godfrey, P.B., and B. Maggs, "The Internet at the speed of light. In Proceedings of the 13th ACM Workshop on Hot Topics in Networks (Oct 2014)", ACM Workshop on Hot Topics in Networks, October 2014,
<<http://speedierweb.web.engr.illinois.edu/cspeed/papers/hotnets14.pdf>>.

[VerfSrc] Vries, W.d., "Verfploeter source code", November 2018,
<<https://github.com/Woutifier/verfploeter>>.

[Vries17b] Vries, W.d., Schmidt, R.d.O., Hardaker, W., Heidemann, J., Boer, P.d., and A. Pras, "Verfploeter - Broad and Load-Aware Anycast Mapping", ACM 2017 Internet Measurement Conference, DOI 10.1145/3131365.3131371, October 2017,
<<https://www.isi.edu/%7ejohnh/PAPERS/Vries17b.pdf>>.

Authors' Addresses

Giovane C. M. Moura
SIDN Labs/TU Delft
Meander 501
6825 MD Arnhem
Netherlands

Phone: +31 26 352 5500
Email: giovane.moura@sidn.nl

Wes Hardaker
USC/Information Sciences Institute
PO Box 382
Davis, 95617-0382
United States of America

Phone: +1 (530) 404-0099
Email: ietf@hardakers.net

John Heidemann
USC/Information Sciences Institute
4676 Admiralty Way
Marina Del Rey, 90292-6695
United States of America

Phone: +1 (310) 448-8708
Email: johnh@isi.edu

Marco Davids
SIDN Labs
Meander 501
6825 MD Arnhem
Netherlands

Phone: +31 26 352 5500
Email: marco.davids@sidn.nl

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 12, 2019

S. Woolf
March 11, 2019

Guidelines for Use of the Special Use Names Registry
draft-stw-6761ext-01

Abstract

RFC 6761 requires that proponents document how a specific name is to be treated within the DNS protocol, public database, and administrative infrastructure, but doesn't provide any guidance to help the community figure out whether a particular registration is otherwise beneficial. This limited guidance in RFC 6761 provides flexibility in standardizing the use of domain names in the modern Internet outside of conventional DNS protocol or the public DNS database. This flexibility has been useful from time to time but has also caused significant confusion (see RFC 8244).

This document attempts to define guidelines for the IESG and the IETF community on the interpretation of RFC 6761 and the use of the special use names registry.

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 September 12, 2019.

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
2. Current SUNR Use Cases	4
3. Guidelines for Special Use Name registration	5
4. The Special Case of Top-Level Domains	6
5. Acknowledgements	8
6. Informative References	8
Author's Address	9

1. Introduction

From time to time, networking protocols need to be able to name things used within the protocol, and resolve the names created or referenced. Such identifiers may also need to be persistent in time, across administrative and operational realms, or through other transformations. Necessary operations tend to include creating, modifying, and deleting names, and accessing values and relationships that correspond to them.

It's common for protocol designers to try to use domain names as the starting point for their systems of names, and the DNS as the starting point for name resolution. This is completely understandable-- domain names, and DNS resolution, are well-established in the expectations of network users and developers, with many advantages in deployment and operation. They're also well-supported by fielded software and a large public database of names and values, with many use cases already represented by example.

However, there are some risks when the protocol designer attempts to re-use domain names and DNS, even (or especially) with modifications, to support a specific use case or protocol design or deployment constraint. These have been touched upon in several RFCs, and in the evolution of DNS protocol itself and the use of domain names as new needs and constraints appear. See in particular RFC 6055 ("IAB Thoughts on Encodings for Internationalized Domain Names"), RFC 6950 ("Architectural Considerations on Application Features in the DNS"),

and RFC 6943 ("Issues in Identifier Comparison for Security Purposes").

Most recently, some of these questions have become prominent in the course of requests for new entries in the special use names registry (or SUNR) as established by RFC 6761 ("Special Use Domain Names"). The topic raises contention in a number of areas, including risks of collision between different authorities and possible confusion among different uses of names within the abstract domain namespace. Issues around the use of the abstract domain namespace have been considered in the DNSOP WG over the last few years and are cataloged in RFC 8244 ("Special-Use Domain Names Problem Statement") at greater length than this document will do.

There are compelling questions that protocol designers or software developers should ask themselves about what behavior they want from the names they use in the context of a new protocol or scope for names. However, rather than boiling that particular ocean, this document attempts the more practical task of providing guidance to the IESG and the community to determine, in broad terms, the benefits and risks of a particular registration in the special use names registry.

RFC 6761 establishes the use of domain names in ways that may be separate to their use in the DNS, but it's somewhat "DNS-centric," in that it doesn't question the default assumption that domain names and DNS-like semantics are desirable or even necessarily acceptable for new naming needs. It also doesn't discuss how one might decide whether a particular string is appropriate for use as a domain name in a particular protocol. The only thing it really requires is a description of how the proposed reserved string should be treated as "special" by DNS resolvers, domain name registrars, and so on.

Primarily RFC 6761 discusses how to make domain names and DNS-like semantics for other networking protocols compatible with the global public DNS. It's left to the protocol designer to decide whether this DNS-centric focus is appropriate for their use case.

Trying to specify how special use domain names interact with the DNS is both necessary for interoperability and helpful in thinking through the proposed "special use". So a proponent of a special use name might discover, in the course of specifying the "special use" for the SUNR, that domain names will not meet the constraints at hand. But even if domain names seem like a good fit for the problem, there's also no guidance in RFC 6761 to deciding what names might or might not be appropriate for the particular need.

The broader discussion of the general applicability of domain names to new needs is useful to consider, and owes a great deal to the RFCs already mentioned, especially RFC 6950, which "provides guidance to application designers and application protocol designers looking to use the DNS to support features in their applications." The consideration there of how to structure domain names and associated data is invaluable. For a different, and sometimes more comprehensive, view on some of the accumulated stresses on the DNS design, see also RFC 8324 ("DNS Privacy, Authorization, Special Uses, Encoding, Characters, Matching, and Root Structure: Time for Another Look?")

This document acknowledges that there may be a need to separate domain names from DNS protocol in the analysis of new protocol needs. For example, RFC 6950 primarily assumes that the namespace, the database of instantiated names, and the protocol for lookup and retrieval are inextricably linked. But more recently, some people are attempting to separate the namespace from specific resolution protocol or even a specific instance of a database of names (namely, the global public Internet DNS). This poses a lot of potential interoperability risk because assumptions about DNS and domain names are so deeply embedded in the internet infrastructure, and it's meeting with varying degrees of drama and varying degrees of success.

Recommended reading on the larger questions includes draft-lewis-domain-names.txt, [RFC1034], [RFC2826], [RFC2860], [RFC6950], [RFC6055], [RFC6943], [RFC6761], [RFC8244] and [RFC8324]. However, this document will consider them out of scope for the immediate problem of providing guidance on the situation we're already in: RFC 6761 is an IETF standards-track document, the special use names registry has been defined, people want to use it, and some uses pose more risk to the interoperability of the Internet than others.

This document is attempting to address the case where the protocol designer believes that something like a domain name is suitable for their protocol, but the use case can't be satisfied by "normal" DNS-- the DNS wire protocol and globally-scoped domain names, resolvable in the public DNS database-- so some additional analysis and specification is needed.

2. Current SUNR Use Cases

Some specific use cases have arisen since the special use names registry was established:

1. Proponents wish to reserve a name to serve a specific purpose in an IETF protocol, discussed as part of protocol definition in an IETF working group. Resolution of the name may be intended for a

limited scope (homenet) or outside of the DNS altogether (mDNS, DNSSD)

2. Proponents wish to reserve a name as used in a protocol developed outside of the IETF, in order to avoid potential collisions with other uses of the namespace. Possible sources of such collisions include future IETF protocols or ICANN's policies for delegation of top-level domains. (.onion, RFC 7686)
 3. Proponents wish to reserve a name from any use in the public DNS, in order to support interoperability and avoid collision or abuse ("localhost," or draft-chapin-additional-reserved-tlds)
3. Guidelines for Special Use Name registration

The use cases and constraints described suggest some specific guidelines for the IESG and the IETF community regarding the use of the special use names registry:

1. Location of a name in the namespace is a consideration. A single-label name or "top level domain" can be attractive at first glance: they can be short and human-friendly, and there's no obvious need to coordinate the use of a top-level label with a TLD operator by, for example, purchasing the use of a second-level domain such as example.org. But the reservation of a TLD also poses a unique challenge, whether the proponent is asking for it to be reserved from use in the DNS root zone, or asking for it to be added to the root zone: the IETF administers the SUNR, but does not control the root zone. Under RFC 2860, ICANN has that authority. More discussion on this point can be seen below, but as a practical matter, IETF Working Groups should not make such requests without compelling justification, and the IESG should not advance them without asking what other options might be available to satisfy relevant technical requirements. (Case: home.arpa, RFC 8375)
1. Compatibility with an installed base might be a compelling need to reserve a specific string as a single label or TLD. This does impose a burden of coordination on ICANN, the IETF, and the IAB, and adds to possible confusion for developers and operators across the wider internet, so the bar to proceeding in this way should be high. There should be significant benefit to interoperability, at the very least. (Case: .onion, .bit, etc.)
2. Preventing ICANN from delegating a name is not, by itself, a compelling reason to reserve it in the SUNR. There's no written policy or agreement that says it would work, and

ICANN may have no process or policy under which it could determine whether such a reservation should be granted. Risking name collision under different policies from different authorities seems unwise, but so does using standards action in one body to constrain policy in another. (Case: home/corp/mail, draft-chapin-additional-tlds)

2. For names reserved as part of an IETF protocol, in a standards-track RFC coming out of an IETF WG, proponents should consider using .arpa (see the IAB note on home.arpa, and RFC 3172). This can work whether the name is supposed to be instantiated in the DNS or not, since the IAB sets policy for .arpa. (Case: home.arpa)
3. Reserved domain names that aren't TLDs require less work for the community because they don't have to be coordinated with another body. All such names, however, should be carefully considered regarding the characteristics discussed above: do they need to exist in the public DNS, or just be valid in a limited scope, or be reserved for another protocol? do they have semantic meaning outside of the specific protocol or scope? do they need to be human-friendly? etc. This may require adding some new questions to the RFC 6761 list, which talks about how the names are treated by DNS but otherwise not much about why they're being reserved or how they're being used. (Case: home.arpa)
4. For names initially reserved or used outside of the IETF, for which a proponent wants to add a special use name registry entry, the bar should be just as high. For single labels in particular, the IESG and the community should require both a stable specification and some assurance that a one-time delegation won't multiply as the protocol evolves or the community forks. This may require a standards-track update to RFC 6761.

4. The Special Case of Top-Level Domains

One key question for all use cases is where in the domain name space a given name should go. This is true regardless of whether the name is intended for resolution in the DNS or as a "protocol switch" to invoke another resolution mechanism.

As noted above, all of the cases described in this document are more difficult if the proponents are attempting to reserve a single label domain name, or "TLD". This is because the IETF delegated authority some time ago to ICANN for the contents of the root zone of the DNS (see RFC 2860).

RFC 6761 claims that the SUNR is based on a "protocol rule" with unchallengeable precedence over ICANN policy. However, it's not clear exactly what this means in practice. There's no process for making a request to ICANN to add a TLD to the root zone, or a string to the list of names ICANN commits won't be delegated, and it seems likely that the effort of inventing one and coordinating it with ICANN would not be justified unless there was a compelling need that couldn't be met any other way.

ICANN has its own community and its own mechanisms for deciding what names should be allowed (or not) in the DNS root zone, and with what constraints. The IETF is not in a position to dictate ICANN's decisions about what names to delegate in the root zone, or even ICANN's policies on what names must not be delegated in the root zone. It can be argued that while ICANN is not an SDO, its relationship to the IETF is not unlike that of an SDO with an overlapping interest in a protocol: while neither can dictate process or policy to the other, an accommodation can generally be found when potential conflicts appear. In the case of the IETF and ICANN, there are several possible mechanisms. The simplest is probably the IETF liaison to the ICANN Board of Directors, for which the IAB appoints the liaison manager (<https://www.iab.org/2018/02/07/call-for-nominations-ietf-liaison-to-icann-board-of-directors-2/>).

In the case of a TLD that the IETF wishes to reserve for "technical use" (per RFC 2860), there's no clear, mutual understanding of what it means. There's also no established guarantee that ICANN won't in the future delegate that name in the public root zone for the DNS. Such a commitment could be requested by the IAB via the IETF liaison or some other means, but there's no assurance it would be obtained, or that the reserved name would be equally useful without such a commitment.

It may also be the case that the IETF wishes ICANN to delegate a TLD in the root zone, with specific characteristics, for "technical use" within the DNS-- such as the requirement seen in discussion of home.arpa, originally specified as .home, that the name should exist in the root zone so that DNSSEC would work as expected in local environments. Again, such a request could be made, but would place an even larger burden on ICANN's policies and processes than a request that they commit to not delegating a name at all. There is no way to project how long it would take or whether it would ultimately succeed.

For these reasons, the bar for the IESG and the IETF community to agree to request a TLD in the SUNR-- either that it should never be delegated, or that it should be delegated according to conditions set by the IETF-- should be very high indeed. The IESG SHOULD NOT make

such requests without a compelling reason that cannot, as a matter of technical necessity, be met by a special use name elsewhere in the domain name space.

5. Acknowledgements

This draft is the outcome of many conversations over many months, including discussions in the DNSOP WG, the IAB, and the ICANN SSAC. Particular thanks to Ed Lewis, Wendy Seltzer, Ralph Droms, Warren Kumari, Lyman Chapin, Dave Thaler, Olaf Kolkman, Brian Trammell, Ted Lemon, David Conrad, Andrew Sullivan, Ted Hardie, John Klensin, and everyone who's expressed exasperation to the author with respect to the issues discussed here.

6. Informative 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>>.
- [RFC2826] Internet Architecture Board, "IAB Technical Comment on the Unique DNS Root", RFC 2826, DOI 10.17487/RFC2826, May 2000, <<https://www.rfc-editor.org/info/rfc2826>>.
- [RFC2860] Carpenter, B., Baker, F., and M. Roberts, "Memorandum of Understanding Concerning the Technical Work of the Internet Assigned Numbers Authority", RFC 2860, DOI 10.17487/RFC2860, June 2000, <<https://www.rfc-editor.org/info/rfc2860>>.
- [RFC2870] Bush, R., Karrenberg, D., Koster, M., and R. Plzak, "Root Name Server Operational Requirements", RFC 2870, DOI 10.17487/RFC2870, June 2000, <<https://www.rfc-editor.org/info/rfc2870>>.
- [RFC6055] Thaler, D., Klensin, J., and S. Cheshire, "IAB Thoughts on Encodings for Internationalized Domain Names", RFC 6055, DOI 10.17487/RFC6055, February 2011, <<https://www.rfc-editor.org/info/rfc6055>>.

- [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>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [RFC6943] Thaler, D., Ed., "Issues in Identifier Comparison for Security Purposes", RFC 6943, DOI 10.17487/RFC6943, May 2013, <<https://www.rfc-editor.org/info/rfc6943>>.
- [RFC6950] Peterson, J., Kolkman, O., Tschafenig, 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>>.
- [RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <<https://www.rfc-editor.org/info/rfc7719>>.
- [RFC8244] Lemon, T., Droms, R., and W. Kumari, "Special-Use Domain Names Problem Statement", RFC 8244, DOI 10.17487/RFC8244, October 2017, <<https://www.rfc-editor.org/info/rfc8244>>.
- [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>>.

Author's Address

Suzanne Woolf
39 Dodge St. #317
Beverly, MA 01915
USA

Email: suzworldwide@gmail.com

DNSOP Working Group
Internet-Draft
Updates: 7873 (if approved)
Intended status: Standards Track
Expires: September 12, 2019

O. Sury
Internet Systems Consortium
W. Toorop
NLnet Labs
March 11, 2019

Algorithms for Domain Name System (DNS) Cookies construction
draft-sury-toorop-dns-cookies-algorithms-00

Abstract

[RFC7873] left the construction of Server Cookies to the discretion of the DNS Server (implementer) which has resulted in a gallimaufry of different implementations. As a result, DNS Cookies are impractical to deploy on multi-vendor anycast networks, because the Server Cookie constructed by one implementation cannot be validated by another.

This document provides precise directions for creating Server Cookies to address this issue. Furthermore, [FNV] is obsoleted as a suitable Hash function for calculating DNS Cookies. [SipHash-2.4] is introduced as a new REQUIRED Hash function for calculating DNS Cookies.

This document updates [RFC7873]

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 September 12, 2019.

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. Contents of this document	3
1.2. Definitions	3
2. Constructing a Client Cookie	3
3. Constructing a Server Cookie	3
3.1. The Version Sub-Field	4
3.2. The Cookie algo Sub-Field	4
3.3. The Reserved Sub-Field	4
3.4. The Timestamp Sub-Field	4
3.5. The Hash Sub-Field	5
4. Cookie Algorithms	5
5. IANA Considerations	6
6. References	6
6.1. Normative References	6
6.2. Informative References	6
Authors' Addresses	7

1. Introduction

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 DNS Cookie implementations, calculating the Server Cookie in different ways. This causes problems with anycast deployments with DNS Software from multiple vendors, because even when all DNS Software would share the same secret, as RECOMMENDED in Section 6. of [RFC7873], they all produce different Server Cookies based on that secret and (at least) the Client Cookie and Client IP Address.

1.1. Contents of this document

In Section 2 instructions for constructing a Client Cookie are given

In Section 3 instructions for constructing a Server Cookie are given

In Section 4 the different hash functions usable for DNS Cookie construction are listed. [FNV] and HMAC-SHA-256-64 [RFC6234] are obsoleted and AES [RFC5649] and [SipHash-2.4] are introduced as a REQUIRED hash function for DNS Cookie implementations.

1.2. Definitions

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

2. Constructing a Client Cookie

The Client Cookie is a nonce and should be treated as such. For simplicity, it can be calculated from Client IP Address, Server IP Address and a secret known only to the Client. The Client Cookie SHOULD have at least 64-bits of entropy. If a secure pseudorandom function (like SipHash24) is used there's no need to change Client secret periodically and change the Client secret only if it has been compromised.

It's recommended but not required that a pseudorandom function is used to construct the Client Cookie:

```
Client-Cookie = MAC_Algorithm(  
    Client IP Address | Server IP Address, Client Secret )
```

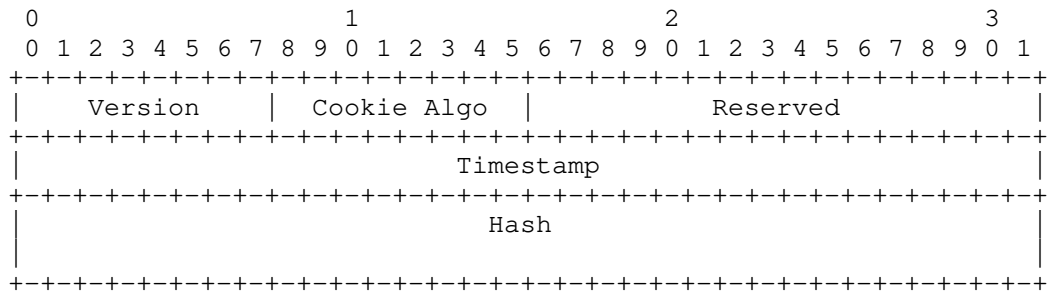
where "|" indicates concatenation.

3. Constructing a Server Cookie

The Server Cookie is effectively message authentication code (MAC) and should be treated as such.

The Server Cookie is not required to be changed periodically if a secure pseudorandom function is used.

The 128-bit Server Cookie consists of Sub-Fields: a 1 octet Version Sub-Field, a 1 octet Cookie Algorithm Sub-Field, a 2 octet Reserved Sub-Field, a 4 octet Timestamp Sub-Field and a 8 octet Hash Sub-Field.



3.1. 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.

3.2. The Cookie algo Sub-Field

The Cookie Algo value defines what algorithm function to use for calculating the Hash Sub-Field as described in Section 3.5. The values are described in Section 4.

3.3. The Reserved Sub-Field

The value of the Reserved Sub-Field is reserved for future versions of Server Side Cookie construction. Even though the value has no specific meaning in this Version, note that it *is* used in determining the Hash value as described in Section 3.5.

3.4. 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 certain time skew between the DNS servers in the anycast.

The DNS Server SHOULD generate new Server Cookie at least if the received Server Cookie from the Client is older than half an hour.

3.5. 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 = Cookie_Algorithm(
    Client Cookie | Version | Cookie Algo | Reserved | TimeStamp,
    Server Secret )
```

4. Cookie Algorithms

Implementation recommendations for Cookie Algorithms [DNSCOOKIE-IANA]:

Number	Mnemonics	Client Cookie	Server Cookie
1	FNV	MUST NOT	MUST NOT
2	HMAC-SHA-256-64	MUST NOT	MUST NOT
3	AES	MAY	MAY
4	SIPHASH24	MUST	MUST

[FNV] is a Non-Cryptographic Hash Algorithm and this document obsoletes the usage of FNV in DNS Cookies.

HMAC-SHA-256-64 is an HMAC-SHA-256 [RFC6234] algorithm reduced to 64-bit. This particular algorithm was implemented in BIND, but it was never the default algorithm and the computational costs makes it unsuitable to be used in DNS Cookies. Therefore this document obsoletes the usage of HMAC-SHA-256 algorithm in the DNS Cookies.

The AES algorithm [RFC5649] has been the default DNS Cookies algorithm in BIND until version x.y.z, and other implementations MAY implement AES algorithm as implemented in BIND for backwards compatibility. However it's recommended that new implementations implement only a pseudorandom functions for DNS Cookies, in this document that would be SipHash24.

[SipHash-2.4] is a pseudorandom function suitable as message authentication code, and this document REQUIRES compliant DNS Server to use SipHash24 as a mandatory and default algorithm for DNS Cookies to ensure interoperability between the DNS Implementations.

5. IANA Considerations

IANA is requested to create and maintain a sub-registry (the "DNS Cookie Algorithm" registry) of the "Domain Name System (DNS) Parameters" registry. The initial values for this registry are described in Section 4.

6. References

6.1. 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>>.
- [RFC5649] Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, DOI 10.17487/RFC5649, September 2009, <<https://www.rfc-editor.org/info/rfc5649>>.
- [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>>.
- [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>>.

6.2. Informative References

- [FNV] Fowler, G., Noll, L., Vo, K., Eastlake, D., and T. Hansen, "The FNV Non-Cryptographic Hash Algorithm", <<https://datatracker.ietf.org/doc/draft-eastlake-fnv>>.
- [SipHash-2.4] Aumasson, J. and D. Bernstein, "SipHash: a fast short-input PRF", 2012, <<https://131002.net/siphash/>>.

Authors' Addresses

Ondrej Sury
Internet Systems Consortium
CZ

Email: ondrej@isc.org

Willem Toorop
NLnet Labs
Science Park 400
Amsterdam 1098 XH
Netherlands

Email: willem@nlnetlabs.nl