

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.

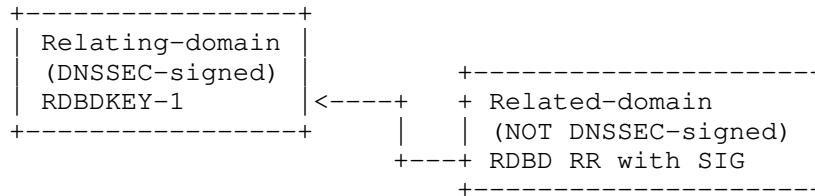
4.2. Extending DNSSEC

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
/bNMQJhDSulcTXxTfZjq0HdzAIR87FcnJ3iegTi1R0iKk/ymRuLGUodNalu+85DB
7XYsy3f/LZoAESasJCWay6kCgYAYGpuc5BvwY5iF5FK/LMVZuH+OuHAF801hI8tg
GI5m/cS7EHD0+aVV38ivYdgRLpowIg4aOCxb19AI2j6KdAbegsgpzyLx5sjmfYBt
1DhgsSyRacFvY0MH3aN289VRCXJxuJeOmqeOaTQHyrX9sN1ctQ+dB/biVvRcrL7q
ziaNQKBgQC9MECoVH/bYJVY6RoC5ZYAa6A4CYDhaXnw40lQ90ckSgWr3FenV7gw
b2xg7zLOX2HZZ+6HejMNGC/efZKVN2Okkpe4KGOXcDH3pYrrkLsLCNRXzxBSyOIt
e3elkAriqiXcr3sPBbn7nakUa7G23O7Hb31C0KGM9f9znN+qWda+3g==
-----END RSA PRIVATE KEY-----
```

my-way.example zone file fragments:


```

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

```

my-way.example private key:

```

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

```

Appendix C. Possible dig output...

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


```
$ dig RDBD my.example

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

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

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

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

Appendix D. Changes and Open Issues

[[RFC editor: please delete this appendix]]

D.1. Changes from -02 to -03

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

D.2. Changes from -01 to -02

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

D.3. Changes from -00 to -01

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

Authors' Addresses

Alex Brotman
Comcast, Inc

Email: alex_brotman@comcast.com

Stephen Farrell
Trinity College Dublin

Email: stephen.farrell@cs.tcd.ie

Network Working Group
Internet-Draft
Updates: RFC 8499 (if approved)
Intended status: Standards Track
Expires: January 26, 2020

P. Hoffman
ICANN
July 25, 2019

Terminology for DNS Transports and Location
draft-hoffman-dns-terminology-ter-02

Abstract

This document adds terms and abbreviations to "DNS Terminology" (RFC 8499) that relate to DNS running over various transports, as well as terms and abbreviations for DNS resolution at traditional and non-traditional locations.

[[This is an early attempt at these terms. They will probably be improved over time.]]

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 26, 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. New Terms and Abbreviations	2
2. Normative References	2
Acknowledgments	3
Author's Address	3

1. New Terms and Abbreviations

The following terms and abbreviations are added to Section 6 of [RFC8499].

DNS-over-TLS (DoT): DNS over TLS as defined in [RFC7858] and its successors.

DNS-over-HTTPS (DoH): DNS over HTTPS as defined in [RFC8484] and its successors.

Classic DNS: DNS over UDP or TCP as defined in [RFC1035] and its successors. "Classic DNS" applies to DNS communication between stub resolvers and recursive resolvers, and between recursive resolvers and authoritative servers.

Recursive DoT (RDoT): RDoT specifically means DNS-over-TLS for transport between stub resolvers and recursive resolvers. This term is necessary because it is expected that DNS-over-TLS will later be defined as a transport between recursive resolvers and authoritative servers,

Authoritative DoT (ADoT): If DNS-over-TLS is later defined as the transport between recursive resolvers and authoritative servers, ADoT specifically means DNS-over-TLS for transport between recursive resolvers and authoritative servers.

2. 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>>.
- [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>>.

Acknowledgments

Sara Dickinson contributed ideas before the first draft was published. Warren Kumari contributed the idea for "Applications Doing DNS" as a term. Many people contributed the idea that it is better to define terms rather than just acronyms.

Author's Address

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

DNSOP Working Group
Internet-Draft
Updates: 6195 (if approved)
Intended status: Standards Track
Expires: January 7, 2020

E. Hunt
D. Mahoney
ISC
July 6, 2019

A DNS Resource Record for Confidential Comments (NOTE RR)
draft-hunt-note-rr-02

Abstract

While the DNS zone master file format has always allowed comments, there is no existing mechanism to preserve comments once the zone has been loaded into memory or converted to a binary representation. This note proposes a new RR type "NOTE", to be allocated from the Covert-RR type range proposed in [I-D.krecicki-dns-covert], so that confidential comments can be stored alongside zone data, and included in zone transfers when Covert semantics are supported by the secondary.

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 7, 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. Definitions	3
2. The NOTE RR Type	3
3. IANA Considerations	3
4. Security and Privacy Considerations	3
5. Normative References	3
Authors' Addresses	4

1. Introduction

DNS zone master files may include comments: any text on a line following an unquoted semicolon is ignored when parsing the file [RFC1034]. These comments are often used by administrators to keep notes about the zone data; for example, the purpose of a particular host, or the person responsible for maintaining it.

When the zone is loaded, however, comments may be lost. Servers which dump backup copies of dynamically updated or automatically signed zones may obliterate comments that were in the original zone files. Secondary servers do not receive comment text when transferring zones from primary servers.

Comments could be stored in the zone itself as TXT RRs; these would be preserved after zone updates and across zone transfers. However, TXT records are available to any DNS query. Because zone file comments commonly include information about internal networks and/or personnel that could be of use to potential attackers, it is better for distribution of comment data to be restricted.

A Covert Resource Record, as described in [I-D.krecicki-dns-covert], could be used for the storage of private text information within zone data itself. This data could be transferred from primary to secondary servers when Covert semantics are supported, and but would be concealed from normal DNS queries (except from specific trusted DNS clients) and from secondary servers that do not signal their support of Covert data transfer.

This document proposes the allocation of a new RR type NOTE from the Covert-RR type range for this purpose. Comments that the operator wishes to be stored and transferred with zone data can be encoded as

NOTE records. Traditional zone file comments, indicated by semicolons, would still be ignored.

1.1. 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. The NOTE RR Type

The NOTE RR is defined for all classes, with mnemonic NOTE and type code (TBD). The RDATA and presentation formats are identical to those of the TXT RR defined in [RFC1035], e.g:

```
$ORIGIN example.com.
joesbox  7200  IN  A      192.0.2.42
7200      IN  AAAA    2001:DB8:3F:B019::17
          0      IN  NOTE  "Desktop system for Joe Smith, x7889"
```

The RR type code MUST be allocated from the Covert-RR type range, and NOTE record data MUST be treated as Covert [I-D.krecicki-dns-covert].

3. IANA Considerations

IANA is requested to allocate a DNS RR type number from the Covert-RR type range for the NOTE RR type.

4. Security and Privacy Considerations

NOTE data should only be accessible via Covert DNS queries, because zone file comments commonly include information that could be of use to potential attackers. Failure to implement the restrictions outlined in [I-D.krecicki-dns-covert] could allow leakage of sensitive information.

5. Normative References

[I-D.krecicki-dns-covert]

Krecicki, W., Hunt, E., and D. Mahoney, "Domain Name System (DNS) Resource Record types for transferring covert information from primary to secondaries", draft-krecicki-dns-covert-00 (work in progress), July 2019.

- [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>>.
- [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>>.

Authors' Addresses

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

Email: each@isc.org

Dan Mahoney
ISC
950 Charter St
Redwood City, CA 94063
US

Email: dmahoney@isc.org

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

DNSOP Working Group
Internet-Draft
Updates: 6195 (if approved)
Intended status: Standards Track
Expires: January 7, 2020

W. Krecicki
E. Hunt
D. Mahoney
ISC
July 6, 2019

Domain Name System (DNS) Resource Record types for transferring covert
information from primary to secondaries
draft-krecicki-dns-covert-00

Abstract

The Domain Name System (DNS) Resource Record TYPES IANA registry reserves the range 128-255 for Q-TYPES and Meta-TYPES [RFC6895] - Resource Records that can only be queried for or contain transient data associated with a particular DNS message.

This document reserves a range of RR TYPE numbers for Covert-TYPES - types that are an integral part of the zone but cannot be accessed via a normal QUERY operation.

Uses for such records could include zone comments that are transferrable with the zone, expiry times for dynamically updated records, or Zone Signing Keys for inline signing. This document, however, does not define any specific Covert RR types.

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 7, 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. Definitions	3
2. Handling of Covert Resource Records	3
2.1. The COVERT-OK option	3
2.2. Protection of Zone Transfers	3
2.3. Authoritative server behaviour	4
2.4. Recursive server behaviour	4
2.5. Interaction with DNSSEC	4
2.6. Interaction with ZONEMD	4
2.7. UPDATE behaviour	4
3. Update to RRTYPE Allocation Template	5
4. Security considerations	5
5. IANA Considerations	5
6. Acknowledgments	5
7. Normative References	5
Authors' Addresses	6

1. Introduction

The Domain Name System (DNS) has no mechanism for sending control information in-band when transferring zone data from primary to secondary servers. This document specifies a range of Resource Record TYPEs that can be used for this purpose. Covert Resource Records can be transferred with the zone during zone transfer, but are not accessible by a normal QUERY operation. It also specifies a method for informing the primary server that the secondary understands Covert semantics, and can be relied upon not to disclose contents of Covert RRs to querying clients.

1.1. 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. Handling of Covert Resource Records

Covert Resource Records require special handling for both queries and zone transfers. This document does not define any specific Covert Resource Record types. When defined, those types may require additional handling on the server side as well; however, that is outside the scope of this document.

2.1. The COVERT-OK option

A client querying or a secondary transferring a zone from a primary server must explicitly signal its understanding of COVERT RR types. The mechanism for this is an EDNS option, with OPTION-CODE [TBD]. OPTION-LENGTH MUST be zero and OPTION-DATA MUST be empty.

2.2. Protection of Zone Transfers

If a secondary server requesting a zone transfer does not understand the Covert semantics, then it will serve the Covert records to its clients. Therefore a protection mechanism must be put in place so that secondary servers that do not understand Covert semantics do not receive Covert records.

If a server requesting a zone transfer understands Covert semantics, it MUST send a COVERT-OK option in the transfer request. If a primary server providing a zone transfer receives such a request, it then knows it can transfer the covert data and the secondary server will cooperate in protecting it.

If the primary server receives a zone transfer request without the COVERT-OK option it MUST NOT transfer the zone with Covert RRs. The default behaviour MUST be to refuse the transfer altogether, but an implementation MAY have a configuration option to allow transfer of the zone with Covert RRs stripped when transferring to a non-compliant secondary.

2.3. Authoritative server behaviour

Covert Resource Records might contain sensitive data; therefore, they MUST NOT be served to regular clients. An authoritative server queried for a Covert RR MUST return an answer as if the RRset the client requested does not exist: NODATA if there are non-Covert Resource Records with the same owner name or the node is an empty non-terminal, or NXDOMAIN otherwise.

The server MAY provide a mechanism allowing clients to query for Resource Records in the Covert range, but it MUST be protected by a mechanism disallowing access from general public (e.g. an ACL or TSIG authentication) and access MUST NOT be enabled by default. The server MUST verify that the query has the COVERT-OK option, and MUST NOT return COVERT records otherwise.

2.4. Recursive server behaviour

Recursive servers MUST NOT send the COVERT-OK option when iterating. If a COVERT record is received in response to an iterative query, it MUST NOT be cached, and it MUST NOT be returned to the client. If a recursive server receives a request for a COVERT record, it MAY iterate to verify whether the answer should be an NXDOMAIN or NODATA, or it MAY simply return a NODATA response immediately.

2.5. Interaction with DNSSEC

Covert Resource Records are not available for regular querying and are used only internally. Their presence in a zone should not, in any way, change the behaviour of that zone for ordinary clients. Therefore, when signing the zone, Covert Resource Records MUST be treated as if they do not exist: - Covert Records MUST NOT be signed. - Nodes that contain only Covert RRs and are not empty non-terminals MUST be omitted from NSEC [RFC4034] and NSEC3 [RFC5155] RR chains. - Any Covert RR types MUST NOT be included in the Type Bit Map field of an NSEC or NSEC3 RR.

2.6. Interaction with ZONEMD

TBD

2.7. UPDATE behaviour

Covert Resource Records MAY be submitted via UPDATE [RFC2136]. Servers SHOULD ignore prerequisites that specify Covert RR types, in order to conceal from untrusted clients the presence or absence of Covert RRs.

3. Update to RRTYPE Allocation Template

The RRTYPE Allocation Template from [RFC6895] is updated to contain a checkbox for Covert-RR:

B.2 Kind of RR: ☐ Data RR ☐ Meta-RR ☐ Covert-RR

4. Security considerations

Since Zone Transfers are unencrypted, the contents of Covert RRs might still be snooped by an on-path attacker. Protection against this kind of attack is outside the scope of this document, but it could be achieved by using, for example, a secure tunnel, a private network, or XFR over TLS transport.

5. IANA Considerations

IANA is requested to reserve range 61440-61695 (0xF000-0xF0FF) in the Resource Record TYPEs registry for Covert types. The procedure for registering RR types from [RFC6895] should be used.

IANA is requested to assign an EDNS option code to the COVERT-OK option.

6. Acknowledgments

TBD

7. 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>>.
- [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>>.
- [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>>.

- [RFC6895] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", BCP 42, RFC 6895, DOI 10.17487/RFC6895, April 2013, <<https://www.rfc-editor.org/info/rfc6895>>.
- [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>>.

Authors' Addresses

Witold Krecicki
ISC
950 Charter St
Redwood City, CA 94063
US

Email: wpk@isc.org

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

Email: each@isc.org

Dan Mahoney
ISC
950 Charter St
Redwood City, CA 94063
US

Email: dmahoney@isc.org

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].

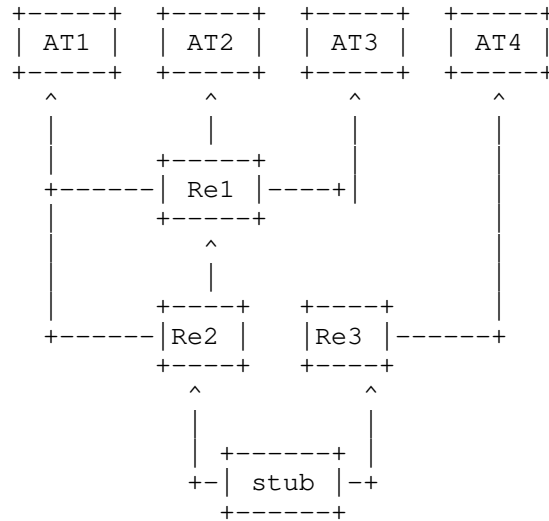


Figure 1: Relationship between recursive resolvers (Re) and authoritative name servers (ATn)

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 choose 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

HTTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2020

B. Schwartz
Google
M. Bishop
E. Nygren
Akamai Technologies
July 8, 2019

HTTPSSVC service location and parameter specification via the DNS (DNS
HTTPSSVC)
draft-nygren-httpbis-httpssvc-03

Abstract

This document specifies an "HTTPSSVC" DNS resource record type to facilitate the lookup of information needed to make connections for HTTPS URIs. The HTTPSSVC DNS RR mechanism allows an HTTPS origin hostname to be served from multiple network services, each with associated parameters (such as transport protocol and keying material for encrypting TLS SNI). It also provides a solution for the inability of the DNS to allow a CNAME to be placed at the apex of a domain name. Finally, it provides a way to indicate that the origin supports HTTPS without having to resort to redirects, allowing clients to remove HTTP from the bootstrapping process.

By allowing this information to be bootstrapped in the DNS, it allows for clients to learn of alternative services before their first contact with the origin. This arrangement offers potential benefits to both performance and privacy.

TO BE REMOVED: This proposal is inspired by and based on recent DNS usage proposals such as ALTSVC, ANAME, and ESNIKEYS (as well as long standing desires to have SRV or a functional equivalent implemented for HTTP). These proposals each provide an important function but are potentially incompatible with each other, such as when an origin is load-balanced across multiple hosting providers (multi-CDN). Furthermore, these each add potential cases for adding additional record lookups in-addition to AAAA/A lookups. This design attempts to provide a unified framework that encompasses the key functionality of these proposals, as well as providing some extensibility for addressing similar future challenges.

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	3
1.1. Introductory Example	4
1.2. Goals of the HTTPSSVC RR	4
1.3. Overview of the HTTPSSVC RR	5
1.4. Additional Alt-Svc parameters	6
1.5. Terminology	6
2. The HTTPSSVC record type	7
2.1. HTTPSSVC RDATA Wire Format	7
2.2. RRNames	8
2.3. SvcRecordType	8
2.4. HTTPSSVC records: alias form	9
2.5. HTTPSSVC records: alternative service form	9
3. Differences from Alt-Svc as transmitted over HTTP	10
3.1. Omitting Max Age and Persist	10
3.2. Multiple records and preference ordering	11
3.3. Constructing Alt-Svc equivalent headers	11
3.4. Granularity and lifetime control	12
4. Client behaviors	12
4.1. Client resolution	12

4.2. HTTP Strict Transport Security	13
4.3. Cache interaction	14
5. DNS Server Behaviors	14
6. Performance optimizations	14
7. Extensions to enhance privacy	15
7.1. Alt-Svc parameter for ESNI keys	15
7.2. Interaction with other standards	15
8. Security Considerations	16
9. IANA Considerations	16
10. Acknowledgements and Related Proposals	17
11. References	17
11.1. Normative References	17
11.2. Informative References	19
Appendix A. Additional examples	20
A.1. Equivalence to Alt-Svc records	20
Appendix B. Comparison with alternatives	20
B.1. Differences from the SRV RRTYPE	20
B.2. Differences from the proposed HTTP record	21
B.3. Differences from the proposed ANAME record	21
B.4. Differences from the proposed ESNI record	21
B.5. SNI Alt-Svc parameter	22
Appendix C. Design Considerations and Open Issues	22
C.1. Record Name	22
C.2. Applicability to other schemes	22
C.3. Wire Format	22
C.4. Extensibility of SvcRecordType	22
C.5. Where to include Priority	22
C.6. Whether to include Weight	23
Appendix D. Change history	23
Authors' Addresses	23

1. Introduction

The HTTPSSVC RR is intended to address a number of challenges facing HTTPS clients and services, while also providing an extensible model to handle similar use-cases without forcing clients to perform additional DNS lookups and without forcing them to first make connections to a default service for the origin.

When clients need to make a connection to fetch resources associated with an HTTPS URI, they must first resolve A and/or AAAA address resource records for the origin hostname. This is adequate when clients default to TCP port 443, do not support Encrypted SNI [ESNI], and where the origin service operator does not have a desire to put an CNAME at a zone apex (such as for "https://example.com"). Handling situations beyond this within the DNS requires learning additional information, and it is highly desirable to minimize the

number of round-trip and lookups required to learn this additional information.

1.1. Introductory Example

As an introductory example, a set of example HTTPSSVC and associated A+AAAA records might be:

```
www.example.com. 2H IN CNAME    svc.example.net.
example.com.     2H IN HTTPSSVC 0 0 svc.example.net.
svc.example.net. 2H IN HTTPSSVC 1 2 svc3.example.net. "h3=\":8003\"; \
                  esnikeys=\"...\""
svc.example.net. 2H IN HTTPSSVC 1 3 svc2.example.net. "h2=\":8002\"; \
                  esnikeys=\"...\""
svc2.example.net. 300 IN A        192.0.2.2
svc2.example.net. 300 IN AAAA     2001:db8::2
svc3.example.net. 300 IN A        192.0.2.3
svc3.example.net. 300 IN AAAA     2001:db8::3
```

In the preceding example, both of the "example.com" and "www.example.com" origin names are aliased to use service endpoints offered as "svc.example.net" (with "www.example.com" continuing to use a CNAME alias). HTTP/2 is available on a cluster of machines located at svc2.example.net with TCP port 8002 and HTTP/3 is available on a cluster of machines located at svc3.example.net with UDP port 8003. An ESNI key is specified which allows the SNI values of "example.com" and "www.example.com" to be encrypted in the handshake with these service endpoints. When connecting, clients will continue to treat the authoritative origins as "https://example.com" and "https://www.example.com", respectively.

1.2. Goals of the HTTPSSVC RR

The goal of the HTTPSSVC RR is to allow clients to resolve a single additional DNS RR in a way that:

- o Provides service endpoints authoritative for an origin, along with parameters associated with each of these endpoints. In particular:
 - * to support connecting directly to [HTTP3] (QUIC transport) service endpoints
 - * to obtain the [ESNI] keys associated with a service endpoint
- o Does not assume that all service endpoints have the same parameters (such as ESNI keys) or capabilities (such as [HTTP3]) or are even operated by the same entity. This is important as DNS

does not provide any way to tie together multiple RRs for the same name. For example, if `www.example.com` is a CNAME alias that switches between one of three CDNs or hosting environments, records (such as A and AAAA) for that name may have been sourced from different environments.

- o Enables the functional equivalent of a CNAME at a zone apex (such as `"example.com"`) for HTTPS traffic, and generally enables delegation of operational authority for an HTTPS origin within the DNS to an alternate name. This addresses a set of long-standing issues due to HTTP(S) clients not implementing support for SRV records, as well as due to a limitation that a DNS name can not have both a CNAME record as well as NS RRs (as is the case for zone apex names)

1.3. Overview of the HTTPSSVC RR

This subsection briefly describes the HTTPSSVC RR in a non-normative manner.

The HTTPSSVC RR has four primary fields:

1. `SvcRecordType`: A numeric flag indicating how to interpret the subsequent fields. When `"0"`, it indicates that the RR contains an alias. When `"1"`, it indicates that the RR contains an alternative service definition.
2. `SvcFieldPriority`: The priority of this record (relative to others, with lower values preferred). Applicable when `SvcRecordType` is `"1"`, and otherwise has value `"0"`. (Described in Section 3.2.)
3. `SvcDomainName`: The domain name of either the alias target (when `SvcRecordType` is `"0"`) or the uri-host domain name of the alternative service endpoint (when `SvcRecordType` is `"1"`).
4. `SvcFieldValue`: An Alternative Service field value describing the alternative service endpoint for the domain name specified in `SvcDomainName` (only when `SvcRecordType` is `"1"` and otherwise empty).

Cooperating DNS recursive resolvers will perform subsequent record resolution (for HTTPSSVC, A, and AAAA records) and return them in the Additional Section of the response. Clients must either use responses included in the additional section returned by the recursive resolver or perform necessary HTTPSSVC, A, and AAAA record resolutions. DNS authoritative servers may attach in-bailiwick

HTTPSSVC, A, AAAA, and CNAME records in the Additional Section to responses for an HTTPSSVC query.

When SvcRecordType is "1", the HTTPSSVC RR extends the concept introduced in the HTTP Alternative Services proposed standard [AltSvc]. Alt-Svc defines:

- o an extensible data model for describing alternative network endpoints that are authoritative for an origin
- o the "Alt-Svc Field Value", a text format for representing this information
- o standards for sending information in this format from a server to a client over HTTP/1.1 and HTTP/2.

Together, these components provide a toolkit that has proven useful and effective for informing a client of alternative services for an origin. However, making use of an alternative service requires contacting the origin server first. This creates an obvious performance cost: users wait for a full HTTP connection initiation (multiple roundtrips) before learning of an alternative service that is preferred by the origin. The first connection also publicly reveals the user's intended destination to all entities along the network path.

The SvcFieldValue includes the Alt-Svc Field Value through the DNS. This is in its standard text format, with the uri-host portion of the alt-authority component moved into the SvcDomainName field of the HTTPSSVC RR. A client receiving this information during DNS resolution can skip the initial connection and proceed directly to an alternative service.

1.4. Additional Alt-Svc parameters

This document also defines one additional Alt-Svc parameter that can be used within SvcFieldValue:

- o esnikeys (Section 7.1): The ESNIKeys structure from Section 4.1 of [ESNI] for use in encrypting the actual origin hostname in the TLS handshake.

1.5. Terminology

For consistency with [AltSvc], we adopt the following definitions:

- o An "origin" is an information source as in [RFC6454].

- o The "origin server" is the server that the client would reach when accessing the origin in the absence of Alt-Svc.
- o An "alternative service" is a different server that can serve the origin.

Abstractly, the origin consists of a scheme (typically "https"), a host name, and a port (typically "443").

Additional DNS terminology intends to be consistent with [DNSTerm].

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. The HTTPSSVC record type

The HTTPSSVC DNS resource record (RR) type (RRTYPE ???) is used to locate endpoints that can service an "https" origin. The presentation format of the record is:

```
RRName TTL Class HTTPSSVC SvcRecordType SvcFieldPriority \
SvcDomainName SvcFieldValue
```

where SvcRecordType is a numeric value of either "0" or "1", SvcFieldPriority is a number in the range 0-65535, SvcDomainName is a domain name, and SvcFieldValue is a string present when SvcRecordType is "1".

The algorithm for resolving HTTPSSVC records and associated address records is specified in Section 4.1.

2.1. HTTPSSVC RDATA Wire Format

The RDATA for the HTTPSSVC RR consists of:

- o a 1 octet flag field for SvcRecordType, interpreted as an unsigned numeric value (0 to 255, with only values "0" and "1" defined here)
- o a 2 octet field for SvcFieldPriority as an integer in network byte order. If SvcRecordType is "0", SvcFieldPriority MUST be 0.
- o the uncompressed SvcDomainName, represented as a sequence of length-prefixed labels as in Section 3.1 of [RFC1035].

- o the SvcFieldValue byte string, consuming the remainder of the record (so smaller than 65535 octets and constrained by the RRDATA and DNS message sizes).

When SvcRecordType is "0", the SvcFieldValue SHOULD be empty ("") and clients MUST ignore the contents of non-empty SvcFieldValue fields.

2.2. RRNames

In the case of the HTTPSSVC RR, an origin is translated into the RRName in the following manner:

1. If the scheme is "https" and the port is 443, then the RRName is equal to the origin host name. Otherwise the RRName is represented by prefixing the port and scheme with "_", then concatenating them with the host name, resulting in a domain name like "_8443._https.www.example.com."
2. When a prior CNAME or HTTPSSVC record has aliased to an HTTPSSVC record, RRName shall be the name of the alias target.

Note that none of these forms alter the HTTPS origin or authority. For example, clients MUST continue to validate TLS certificate hostnames based on the origin host.

As an example for schemes and ports other than "https" and port 443:

```
_8443._wss.api.example.com. 2H IN HTTPSSVC 0 0 svc4.example.net.  
svc4.example.net. 2H IN HTTPSSVC 1 3 svc4.example.net. "h2=\":8004\"; \\  
esnikeys=\\\"...\\\""
```

would indicate that "wss://api.example.com:8443" is aliased to use HTTP/2 service endpoints offered as "svc4.example.net" on port 8004.

2.3. SvcRecordType

The SvcRecordType field is a numeric value defined to be either "0" or "1". Within an HTTPSSVC RRSet, all RRs must have the same value for SvcRecordType. Clients and recursive servers MUST ignore HTTPSSVC resource records with other SvcRecordType values. If an RRSet contains a record with type "0", the client MUST ignore any records in the set with type "1".

When SvcRecordType is "0", the HTTPSSVC is defined to be in "alias form".

When SvcRecordType is "1", the HTTPSSVC is defined to be in "alternative service form".

2.4. HTTPSSVC records: alias form

When `SvcRecordType` is "0", the HTTPSSVC record is to be treated similar to a CNAME alias pointing to the domain name specified in `SvcDomainName`. HTTPSSVC RRsets MUST only have a single resource record in this form. If multiple are present, clients or recursive resolvers SHOULD pick one non-deterministically.

The common use-case for this form of the HTTPSSVC record is as an alternative to CNAMEs at the zone apex where they are not allowed. For example, if an operator of `https://example.com` wanted to point HTTPS requests to a service operating at `svc.example.net`, they would publish a record such as:

```
example.com. 3600 IN HTTPSSVC 0 0 svc.example.net.
```

The `SvcDomainName` MUST point to a domain name that contains another HTTPSSVC record and/or address (AAAA and/or A) records.

Note that the `RRName` and the `SvcDomainName` MAY themselves be CNAMEs. Clients and recursive resolvers MUST follow CNAMEs as normal.

Due to the risk of loops, clients and recursive resolvers MUST implement loop detection. Chains of consecutive HTTPSSVC and CNAME records SHOULD be limited to (8?) prior to reaching terminal address records.

The `SvcFieldValue` in this form SHOULD be an empty string and clients MUST ignore its contents.

As legacy clients will not know to use this record, service operators will likely need to retain fallback AAAA and A records alongside this HTTPSSVC record, although in a common case the target of the HTTPSSVC record might have better performance, and therefore would be preferable for clients implementing this specification to use.

2.5. HTTPSSVC records: alternative service form

When `SvcRecordType` is "1", the combination of `SvcDomainName` and `SvcFieldValue` within each resource record associates an Alternative Service Field Value with an origin.

The `SvcFieldValue` of the HTTPSSVC resource record contains an Alt-Svc Field Value, exactly as defined in Section 4 of [AltSvc], but with the `uri-host` moved to the `SvcDomainName` field.

For example, if the operator of `https://www.example.com` intends to include an HTTP response header like


```
Alt-Svc: h3="svc.example.net:8003"; ma=3600, \
        h2="svc.example.net:8002"; ma=3600
```

they could also publish an HTTPSSVC DNS RRSet like

```
www.example.com. 3600 IN HTTPSSVC 1 2 svc.example.net. "h3=\":8003\""  
                    HTTPSSVC 1 3 svc.example.net. "h2=\":8002\""
```

This data type can be represented as an Unknown RR as described in [RFC3597]:

```
www.example.com. 3600 IN TYPE??? \\\# TBD:WRITE ME
```

This construction is intended to be extensible in two ways. First, any extensions that are made to the Alt-Svc format for transmission over HTTPS are also applicable here, unless expressly mentioned otherwise.

Second, by defining a way to map non-HTTPS schemes and non-default ports (Section 2.2), we provide a way for the HTTPSSVC to be used for them as needed. However, by using the origin name for the RRName for scheme https and port 443 we allow HTTPSSVC records to be included at the end of CNAME chains for existing site implementations without requiring changes in the zone containing the origin.

3. Differences from Alt-Svc as transmitted over HTTP

Publishing an alternative services form HTTPSSVC record in DNS is intended to be equivalent to transmitting this field value over HTTPS, and receiving an HTTPSSVC record is intended to be equivalent to receiving this field value over HTTPS. However, there are some small differences in the intended client and server behavior.

3.1. Omitting Max Age and Persist

When publishing an HTTPSSVC record in DNS, server operators **MUST** omit the "ma" parameter, which encodes the "max age" (i.e. expiration time) of an Alt-Svc Field Value. Instead, server operators **SHOULD** encode the expiration time in the DNS TTL, and **MUST NOT** set a TTL longer than the intended "max age".

When receiving an HTTPSSVC record, clients **SHOULD** synthesize a new "ma" parameter from the DNS TTL if the resulting alt-value is being passed to a subsystem that might employ caching.

When publishing an HTTPSSVC record, server operators **MUST** omit the "persist" parameter, which indicates whether the client should use this record on other network paths. When receiving an HTTPSSVC

record, clients MUST discard any records that contain a "persist" flag. Disabling persistence is important to prevent a local adversary in one network from implanting a forged DNS record that allows them to track users or hinder their connections after they leave that network.

3.2. Multiple records and preference ordering

Server operators MAY publish multiple SvcRecordType "1" HTTPSSVC records as an RRSET. When converting a collection of alt-values into an HTTPSSVC RRSET, the server operator MUST set the overall TTL to a value no larger than the minimum of the "max age" values (following Section 5.2 of [RFC2181]).

Each RR MUST contain exactly one alt-value, as described in Section 3 of [AltSvc].

As RRs within an RRSET are explicitly unordered collections, the SvcFieldPriority value is introduced to indicate priority. HTTPSSVC RRs with a smaller SvcFieldPriority value SHOULD be given preference over RRs with a larger SvcFieldPriority value.

Alt-values received via HTTPS are preferred over any Alt-value received via DNS.

When receiving an RRSET containing multiple HTTPSSVC records with the same SvcFieldPriority value, clients SHOULD apply a random shuffle within a priority level to the records before using them, to ensure randomized load-balancing.

3.3. Constructing Alt-Svc equivalent headers

For a client to construct the equivalent of an Alt-Svc HTTP response header:

1. For each RR, the SvcDomainName MUST be inserted as the uri-host. If SvcDomainName is has the value "." then the RRNAME for the final HTTPSSVC record MUST be inserted as the uri-host. (In the case of a CNAME or a HTTPSSVC SvcRecordType "0" record pointing to an HTTPSSVC record with SvcRecordType "1" and SvcDomainName "." then it is the RRNAME for the terminal HTTPSSVC record that must be inserted as the uri-host.)
2. The RRs SHOULD be ordered by increasing SvcFieldPriority, with shuffling for equal SvcFieldPriority values. Clients MAY choose to further prioritize alt-values where address records are immediately available for the alt-value's SvcDomainName.

3. The client SHOULD concatenate the thus-transformed-and-ordered SvcFieldValues in the RRSET, separated by commas. (This is semantically equivalent to receiving multiple Alt-Svc HTTP response headers, according to Section 3.2.2 of [HTTP]).

3.4. Granularity and lifetime control

Sending Alt-Svc over HTTP allows the server to tailor the Alt-Svc Field Value specifically to the client. When using an HTTPSSVC DNS record, groups of clients will necessarily receive the same Alt-Svc Field Value. Therefore, this standard is not suitable for uses that require single-client granularity in Alt-Svc.

Some DNS caching systems incorrectly extend the lifetime of DNS records beyond the stated TTL. Server operators MUST NOT rely on HTTPSSVC records expiring on time, and MAY shorten the TTL to compensate.

4. Client behaviors

4.1. Client resolution

When attempting to resolve a name HOST, clients should follow in-order:

1. Issue parallel AAAA/A and HTTPSSVC queries for the name HOST. The answers for these may or may not include CNAME pointers before reaching one or more of these records.
2. If an HTTPSSVC record of SvcRecordType "0" is returned for HOST, clients should loop back to step 1 replacing HOST with SvcDomainName, subject to loop detection heuristics.
3. If one or more HTTPSSVC record of SvcRecordType "1" is returned for HOST, clients should synthesize equivalent Alt-Svc Field Values based on the SvcDomainName and SvcFieldValue. If one of these alt-values is selected to be used in a connection, the client will need to resolve AAAA and/or A records for SvcDomainName.
4. If only AAAA and/or A records are present for HOST (and no HTTPSSVC), clients should make a connection to one of the IP addresses contained in these records and proceed normally.

When selecting between AAAA and A records to use, clients may use an approach such as [HappyEyeballsV2]

Some possible optimizations are discussed in Section 6 to reduce latency impact in comparison to ordinary AAAA/A lookups.

4.2. HTTP Strict Transport Security

By publishing an HTTPSSVC record, the server operator indicates that all useful HTTP resources on that origin are reachable over HTTPS, similar to HTTP Strict Transport Security [HSTS]. When an HTTPSSVC record is present for an origin, all "http" scheme requests for that origin SHOULD logically be redirected to "https".

Prior to making an "http" scheme request, the client SHOULD perform a lookup to determine if an HTTPSSVC record is available for that origin. To do so, the client SHOULD construct a corresponding "https" URL as follows:

1. Replace the "http" scheme with "https".
2. If the "http" URL explicitly specifies port 80, specify port 443.
3. Do not alter any other aspect of the URL.

This construction is equivalent to Section 8.3 of [HSTS] , point 5.

If an HTTPSSVC record is present for this "https" URL, the client should treat this as the equivalent of receiving an HTTP "307 Temporary Redirect" redirect to the "https" URL. Because HTTPSSVC is received over an often insecure channel (DNS), clients MUST NOT place any more trust in this signal than if they had received a 307 redirect over cleartext HTTP.

If the HTTPSSVC query results in a SERVFAIL error, and the connection between the client and the recursive resolver is cryptographically protected (e.g. using TLS [RFC7858] or HTTPS [RFC8484]), the client SHOULD abandon the connection attempt and display an error message. A SERVFAIL error can occur if the domain is DNSSEC-signed, the recursive resolver is DNSSEC-validating, and an active attacker between the recursive resolver and the authoritative DNS server is attempting to prevent the upgrade to HTTPS.

Similarly, if the client enforces DNSSEC validation on A/AAAA RRs, it SHOULD abandon the connection attempt if the HTTPSSVC RR fails to validate.

4.3. Cache interaction

If the client has an Alt-Svc cache, and a usable Alt-Svc value is present in that cache, then the client SHOULD NOT issue an HTTPSSVC DNS query. Instead, the client SHOULD proceed with alternative service connection as usual.

If the client has a cached Alt-Svc entry that is expiring, the client MAY perform an HTTPSSVC query to refresh the entry.

5. DNS Server Behaviors

Recursive DNS servers SHOULD resolve SvcDomainName records and include them in the Additional Section (along with any relevant CNAME records). For SvcRecordType=0, recursive DNS servers SHOULD attempt to resolve and include A, AAAA, and HTTPSSVC records. For SvcRecordType=1, recursive DNS servers SHOULD attempt to resolve and include A and AAAA records.

Authoritative DNS servers SHOULD return A, AAAA, and HTTPSSVC records (as well as any relevant CNAME records) in the Additional Section for any in-bailiwick SvcDomainNames.

6. Performance optimizations

For optimal performance (i.e. minimum connection setup time), clients SHOULD issue address (AAAA and/or A) and HTTPSSVC queries simultaneously, and SHOULD implement a client-side DNS cache. With these optimizations in place, and conforming DNS servers, using HTTPSSVC does not add network latency to connection setup.

A nonconforming recursive resolver might return an HTTPSSVC response with a nonempty SvcDomainName, without the corresponding address records. If all the HTTPSSVC RRs in the response have nonempty SvcDomainName values, and the client does not have address records for any of these values in its DNS cache, the client SHOULD perform an additional address query for the selected SvcDomainName.

The additional DNS query in this case introduces a delay. To avoid causing a delay for clients using a nonconforming recursive resolver, domain owners SHOULD choose the SvcDomainName to be a name in the origin hostname's CNAME chain if possible. This will ensure that the required address records are already present in the client's DNS cache as part of the responses to the address queries that were issued in parallel.

Highly performance-sensitive clients MAY implement the following special- case shortcut to avoid increased connection time: if (1) one

of the HTTPSSVC records returned has `SvcRecordType=0`, (2) its `SvcDomainName` is not in the DNS cache, and (3) the address queries for the origin domain return usable IP addresses, then the client MAY ignore the HTTPSSVC records and connect directly to the origin domain. When the `SvcDomainNames` and any needed HTTPSSVC records are available, the client SHOULD make subsequent requests over connections specified by the HTTPSSVC records.

Server operators can therefore expect that publishing HTTPSSVC records with `SvcRecordType=0` should not cause an additional DNS query for performance-sensitive clients. Server operators who wish to prevent this optimization should use `SvcRecordType=1`.

7. Extensions to enhance privacy

7.1. Alt-Svc parameter for ESNI keys

An Alt-Svc "esnikeys" parameter is defined for specifying ESNI keys corresponding to an alternative service. The value of the parameter is an `ESNIKeys` structure [ESNI] encoded in [base64], or the empty string. ESNI-aware clients SHOULD prefer alt-values with nonempty esnikeys.

This parameter MAY also be sent in Alt-Svc HTTP response headers and HTTP/2 ALTSVC frames.

The Alt-Svc specification states that "the client MAY fall back to using the origin" in case of connection failure [AltSvc]. This behavior is not suitable for ESNI, because fallback would negate the privacy benefits of ESNI.

Accordingly, any connection attempt that uses ESNI MUST fall back only to another alt-value that also has the esnikeys parameter. If the parameter's value is the empty string, the client SHOULD connect as it would in the absence of any `ESNIKeys` information.

For example, suppose a server operator has two alternatives. Alternative A is reliably accessible but does not support ESNI. Alternative B supports ESNI but is not reliably accessible. The server operator could include a full esnikeys value in Alternative B, and mark Alternative A with `esnikeys=""` to indicate that fallback from B to A is allowed.

7.2. Interaction with other standards

The purpose of this standard is to reduce connection latency and improve user privacy. Server operators implementing this standard SHOULD also implement TLS 1.3 [RFC8446] and OCSP Stapling [RFC6066],

both of which confer substantial performance and privacy benefits when used in combination with HTTPSSVC records.

To realize the greatest privacy benefits, this proposal is intended for use with a privacy-preserving DNS transport (like DNS over TLS [RFC7858] or DNS over HTTPS [RFC8484]). However, performance improvements, and some modest privacy improvements, are possible without the use of those standards.

This RRTYPE could be extended to support schemes other than "https". Any such scheme MUST have an entry under the HTTPSSVC RRTYPE in the IANA DNS Underscore Global Scoped Entry Registry [Attrleaf]. The scheme SHOULD have an entry in the IANA URI Schemes Registry [RFC7595]. The scheme SHOULD be one for which Alt-Svc is defined.

8. Security Considerations

Alt-Svc Field Values are intended for distribution over untrusted channels, and clients are REQUIRED to verify that the alternative service is authoritative for the origin (Section 2.1 of [AltSvc]). Therefore, DNSSEC signing and validation are OPTIONAL for publishing and using HTTPSSVC records.

TBD: expand this section in more detail. In particular: * Just as with [AltSvc], clients must validate the TLS server certificate against hostname associated with the origin. Clients MUST NOT use the SvcDomainName as any part of the server TLS certificate validation. * ...

9. IANA Considerations

Per [RFC6895], please add the following entry to the data type range of the Resource Record (RR) TYPEs registry:

TYPE	Meaning	Reference
HTTPSSVC	HTTPS Service Location	(This document)

Per [Attrleaf], please add the following entries to the DNS Underscore Global Scoped Entry Registry:

RR TYPE	_NODE NAME	Meaning	Reference
HTTPSSVC	_https	Alt-Svc for HTTPS	(This document)

Per [AltSvc], please add the following entries to the HTTP Alt-Svc Parameter Registry:

Alt-Svc Parameter	Meaning	Reference
esnikeys	Encrypted SNI keys	(This document)

10. Acknowledgements and Related Proposals

There have been a wide range of proposed solutions over the years to the "CNAME at the Zone Apex" challenge proposed. These include [I-D.draft-bellis-dnsop-http-record-00], [I-D.draft-ietf-dnsop-aname-03], and others.

Thank you to Ian Swett, Ralf Weber, Jon Reed, Martin Thompson, Lucas Pardue, Ilari Liusvaara, and others for their feedback and suggestions on this draft.

11. References

11.1. Normative References

- [AltSvc] Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <<https://www.rfc-editor.org/info/rfc7838>>.
- [AltSvcSNI] Bishop, M., "The "SNI" Alt-Svc Parameter", draft-bishop-httpbis-sni-altsvc-02 (work in progress), May 2018.
- [Attrleaf] Crocker, D., "DNS Scoped Data Through "Underscore" Naming of Attribute Leaves", draft-ietf-dnsop-attrleaf-16 (work in progress), November 2018.
- [base64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [ESNI] Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "Encrypted Server Name Indication for TLS 1.3", draft-ietf-tls-esni-03 (work in progress), March 2019.

- [HappyEyeballsV2] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <<https://www.rfc-editor.org/info/rfc8305>>.
- [HSTS] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, DOI 10.17487/RFC6797, November 2012, <<https://www.rfc-editor.org/info/rfc6797>>.
- [HTTP3] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", draft-ietf-quic-http-20 (work in progress), April 2019.
- [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>>.
- [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>>.
- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/info/rfc6066>>.
- [RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <<https://www.rfc-editor.org/info/rfc6454>>.
- [RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <<https://www.rfc-editor.org/info/rfc7595>>.

- [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>>.
- [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>>.
- [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>>.
- [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>>.

11.2. Informative References

- [DNSTerm] 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>>.
- [HTTP] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [I-D.draft-bellis-dnsop-http-record-00] Bellis, R., "A DNS Resource Record for HTTP", draft-bellis-dnsop-http-record-00 (work in progress), November 2018.
- [I-D.draft-ietf-dnsop-aname-03] Finch, T., Hunt, E., Dijk, P., Eden, A., and W. Mekking, "Address-specific DNS aliases (ANAME)", draft-ietf-dnsop-aname-03 (work in progress), April 2019.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <<https://www.rfc-editor.org/info/rfc2782>>.
- [RFC6895] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", BCP 42, RFC 6895, DOI 10.17487/RFC6895, April 2013, <<https://www.rfc-editor.org/info/rfc6895>>.

Appendix A. Additional examples

A.1. Equivalence to Alt-Svc records

The following:

```
www.example.com. 2H IN CNAME    svc.example.net.
example.com.     2H IN HTTPSSVC 0 0 svc.example.net.
svc.example.net. 2H IN HTTPSSVC 1 2 svc3.example.net. "h3=\":8003\"; \
                  esnikeys=\"ABC...\""
svc.example.net. 2H IN HTTPSSVC 1 3 . "h2=\":8002\"; \
                  esnikeys=\"123...\""
```

is equivalent to the Alt-Svc record:

```
Alt-Svc: h3="svc3.example.net:8003"; esnikeys="ABC..."; ma=7200, \
         h2="svc.example.net:8002"; esnikeys="123..."; ma=7200
```

for the origins of both "https://www.example.com" and
"https://example.com".

Appendix B. Comparison with alternatives

The HTTPSSVC record type closely resembles some existing record types and proposals. A complaint with all of the alternatives is that web clients have seemed unenthusiastic about implementing them. The hope here is that by providing an extensible solution that solves multiple problems we will overcome the inertia and have a path to achieve client implementation.

B.1. Differences from the SRV RRTYPE

An SRV record [RFC2782] can perform a similar function to the HTTPSSVC record, informing a client to look in a different location for a service. However, there are several differences:

- o SRV records are typically mandatory, whereas clients will always continue to function correctly without making use of Alt-Svc or HTTPSSVC.
- o SRV records cannot instruct the client to switch or upgrade protocols, whereas Alt-Svc can signal such an upgrade (e.g. to HTTP/2).
- o SRV records are not extensible, whereas Alt-Svc and thus HTTPSSVC can be extended with new parameters. For example, this is what allows the incorporation of ESNI keys in HTTPSSVC.

- o Using SRV records would not allow a client to skip processing of the Alt-Svc information in a subsequent connection, so it does not confer a performance advantage.

B.2. Differences from the proposed HTTP record

Unlike [I-D.draft-bellis-dnsop-http-record-00], this approach is extensible to cover Alt-Svc and ESNIKeys use-cases. Like that proposal, this addresses the zone apex CNAME challenge.

Like that proposal it remains necessary to continue to include address records at the zone apex for legacy clients.

B.3. Differences from the proposed ANAME record

Unlike [I-D.draft-ietf-dnsop-aname-03], this approach is extensible to cover Alt-Svc and ESNIKeys use-cases. This approach also does not require any changes or special handling on either authoritative or master servers, beyond optionally returning in-bailiwick additional records.

Like that proposal, this addresses the zone apex CNAME challenge for clients that implement this.

However with this HTTPSSVC proposal it remains necessary to continue to include address records at the zone apex for legacy clients. If deployment of this standard is successful, the number of legacy clients will fall over time. As the number of legacy clients declines, the operational effort required to serve these users without the benefit of HTTPSSVC indirection should fall. Server operators can easily observe how much traffic reaches this legacy endpoint, and may remove the apex's address records if the observed legacy traffic has fallen to negligible levels.

B.4. Differences from the proposed ESNI record

Unlike [ESNI], this approach is extensible and covers the Alt-Svc case as well as addresses the zone apex CNAME challenge.

By using the Alt-Svc model we also provide a way to solve the ESNI multi-CDN challenges in a general case.

Unlike ESNI, this is focused on the specific case of HTTPS, although this approach could be extended for other protocols. It also allows specifying ESNI keys for a specific port, not an entire host.

B.5. SNI Alt-Svc parameter

Defining an Alt-Svc sni= parameter (such as from [AltSvcSNI]) would have provided some benefits to clients and servers not implementing ESNI, such as for specifying that "_wildcard.example.com" could be sent as an SNI value rather than the full name. There is nothing precluding HTTPSSVC from being used with an sni= parameter if one were to be defined, but it is not included here to reduce scope, complexity, and additional potential security and tracking risks.

Appendix C. Design Considerations and Open Issues

This draft is intended to be a work-in-progress for discussion. Many details are expected to change with subsequent refinement. Some known issues or topics for discussion are listed below.

C.1. Record Name

Naming is hard. The "HTTPSSVC" is proposed as a placeholder. Other names for this record might include ALTSVC, HTTPS, HTTPSSRV, B, or something else.

C.2. Applicability to other schemes

The focus of this record is on optimizing the common case of the "https" scheme. It is worth discussing whether this is a valid assumption or if a more general solution is applicable. Past efforts to over-generalize have not met with broad success.

C.3. Wire Format

Advice from experts in DNS wire format best practices would be greatly appreciated to refine the proposed details, overall.

C.4. Extensibility of SvcRecordType

Only values of "0" and "1" are allowed for SvcRecordType. Should we give more thought to potential future values? The current version tries to leave this open by indicating that resource records with unknown SvcRecordType values should be ignored (and perhaps should be switched to MUST be ignored)?

C.5. Where to include Priority

The SvcFieldPriority could alternately be included as a pri= Alt-Svc attribute. It wouldn't be applicable for Alt-Svc returned via HTTP, but it is also not necessarily needed by DNS servers. It is also not used when SvcRecordType=0. A related question is whether to omit it

from the textual representation when SvcRecordType=0. Regardless, having a series of sequential numeric values in the textual representation has risk of user error, especially as MX, SRV, and others all have their own variations here.

C.6. Whether to include Weight

Some other similar mechanisms such as SRV have a weight in-addition to priority. That is excluded here for simplicity. It could always be added as an optional Alt-Svc attribute.

Appendix D. Change history

- o draft-nygren-httpbis-httpssvc-03
 - * Change redirect type for HSTS-style behavior from 302 to 307 to reduce ambiguities.
- o draft-nygren-httpbis-httpssvc-02
 - * Remove the redundant length fields from the wire format.
 - * Define a SvcDomainName of "." for SvcRecordType=1 as being the HTTPSSVC RRNAME.
 - * Replace "hq" with "h3".
- o draft-nygren-httpbis-httpssvc-01
 - * Fixes of record name. Replace references to "HTTPSVC" with "HTTPSSVC".
- o draft-nygren-httpbis-httpssvc-00
 - * Initial version

Authors' Addresses

Ben Schwartz
Google

Email: bemasc@google.com

Mike Bishop
Akamai Technologies

Email: mbishop@evequefou.be

Erik Nygren
Akamai Technologies

Email: erik+iETF@nygren.org

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

P. Sood
Google
R. Arends
P. Hoffman
ICANN
June 27, 2019

DNS Resolver Information Self-publication
draft-sah-resolver-information-02

Abstract

This document describes methods for DNS resolvers to self-publish information about themselves, such as whether they perform DNSSEC validation or are available over transports other than what is defined in RFC 1035. The information is returned as a JSON object. The names in this object are defined in an IANA registry that allows for light-weight registration. Applications and operating systems can use the methods defined here to get the information from resolvers in order to make choices about how to send future queries to those resolvers.

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
1.1. Definitions	3
2. Retrieving Resolver Information by DNS	3
3. Retrieving Resolver Information by Well-Known URI	4
4. Contents of the Returned I-JSON Object	5
4.1. The "inventory" name	6
4.2. Example	6
5. IANA Considerations	6
5.1. RESINFO RRtype	6
5.2. Registry for DNS Resolver Information	6
5.3. resolver-info Well-known URI	7
6. Security Considerations	7
7. References	8
7.1. Normative References	8
7.2. Informative References	8
Acknowledgments	9
Authors' Addresses	9

1. Introduction

Historically, DNS stub resolvers typically communicated with the recursive resolvers in their configuration without needing to know anything about the features of the recursive resolvers. More recently, recursive resolvers have different features that may cause stub resolvers to make choices about which configured resolver from its configuration to use, and also how to communicate with the recursive resolver (such as over different transports). Thus stub resolvers need a way to get information from recursive resolvers about features that might affect the communication.

This document specifies methods for stub resolvers to ask recursive resolvers for such information. In short, a new RRtype is defined for stub resolvers to query using the DNS, and a new well-known URI is defined for stub resolvers to query using HTTP over TLS.

The response from either method is the same: a JSON object. The JSON object MUST use the I-JSON message format defined in [RFC7493]. Note that [RFC7493] was based on RFC 7159, but RFC 7159 was replaced by

[RFC8259]. Requiring the use of I-JSON instead of more general JSON format greatly increases the likelihood of interoperability.

The information that a resolver might want to give to a recursive resolver is not defined in this document; instead other documents will follow that will specify that information and the format that it comes in.

It is important to note that the protocol defined here is only for recursive resolvers, not for authoritative servers. Authoritative servers MUST NOT answer queries that are defined in this protocol. (It is likely that a later protocol will allow authoritative servers to give information in a method similar to the one described in this document.)

1.1. Definitions

In the rest of this document, the term "resolver" without qualification means "recursive resolver" as defined in [RFC8499]. Also, the term "stub" is used to mean "stub resolver".

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. Retrieving Resolver Information by DNS

A stub that wants to use the DNS to get information about a resolver can use the DNS query defined here. The query a stub resolver uses is <reverse-ip>.{in-addr,ip6}.arpa/IN/RESINFO. The RRtype "RESINFO" is defined in this document, and the IANA assignment is given in Section 5.1. The contents of the Rdata in the response to this query is defined in Section 4. If the resolver understands the RESINFO RRtype, the RRset in the Answer section MUST have exactly one record.

In this section, "<reverse-ip>.{in-addr,ip6}.arpa" is the domain name associated with the reverse lookup of an IP address of the resolver (resolvers can have multiple addresses). For example, if the resolver is at 192.0.2.1, the query would be 1.2.0.192.in-addr.arpa/IN/RESINFO.

A resolver that receives a query with the RRtype of RESINFO with a QNAME of <reverse-ip>.{in-addr,ip6}.arpa acts as if it is delegated, and responds with its own RESINFO data in the Answer section. The resolver can generate this reply with special code to capture queries for these types of addresses; if the resolver can be configured to

also be authoritative for some zones, it can use that configuration to actually be authoritative for the addresses on which it responds.

A stub that knows a specific type of information it wants MAY ask for that information by prepending a label with the name of the information in its query. For example, if the stub knows that it wants information whose name is "temp-field2", it would send the query temp-field2.<reverse-ip>.{in-addr,ip6}.arpa/IN/RESINFO. As described in Section 4, the JSON object in the response is likely to have name/value pairs in addition to the one requested.

Any query for the RESINFO RRtype that is not in <reverse-ip>.{in-addr,ip6}.arpa/IN or a subdomain of <reverse-ip>.{in-addr,ip6}.arpa/IN is meaningless and MUST result in a NODATA or NXDOMAIN response. Resolvers would not need any special code to meet this requirement; they only need code to handle the RESINFO RRtype that is not in <reverse-ip>.{in-addr,ip6}.arpa/IN or a subdomain of <reverse-ip>.{in-addr,ip6}.arpa/IN .

3. Retrieving Resolver Information by Well-Known URI

A stub that wants to use HTTPS to get information about a resolver can use the well-known URI defined here. Because this uses HTTPS, the stub has the possibility of authenticating the TLS connection. If the connection cannot be authenticated (such as if the stub only knows the IP address of the resolver and the resolver's certificate does not have the IP address, or the correct IP address), the stub MAY still use the results with the same lack of assuredness as it would have with using a DNS request described in Section 2.

The stub MUST use the HTTP GET method. The URI used to get the resolver information is one of:

`https://IPADDRESSOFRESOLVER/.well-known/resolver-info/`

`https://DOMAINNAMEOFRESOLVER/.well-known/resolver-info/`

This uses the ".well-known" URI mechanism defined in [RFC8615]. The contents of the response to this query is defined in Section 4.

A resolver that uses this protocol to publish its information SHOULD, if possible, have a TLS certificate whose subject identifiers are any IP address that the resolver is available on, as well as any domain names that the resolver operator uses for the resolver. At the time that this document is published, getting IP addresses in TLS certificates is possible, but there are only a few widely-trusted CAs that issue such certificates. [I-D.ietf-acme-ip] describes a new

protocol that may cause IP address certificates to become more common.

In the future, DHCP and/or DHCPv6 and/or RA may have options that allow the configuration to contain the domain name of a resolver. If so, this can be used for matching the domain name in the TLS certificate.

4. Contents of the Returned I-JSON Object

The JSON object returned by a DNS query or an HTTPS query MUST contain at least one name/value pair: "inventory", described later in this section. The returned object MAY contain any other name/value pairs.

The requirement for the inclusion of the "inventory" name/value pair is so that systems retrieving the information over DNS can create specific queries. Using specific queries can reduce the number of round trips in the case where the answers to queries become large. The "inventory" name/value pair MUST be included in the response even if the query was for a single name.

If the request was over DNS using a subdomain under <reverse-ip>.{in-addr,ip6}.arpa, the resolver SHOULD return an object that contains a name/value pair with that name if the resolver has that information. If the resolver does not have information for that name, it MUST NOT return the name in the object.

If the request was over HTTPS, the resolver SHOULD return an object with all known name/value pairs for which it has information.

All names in the returned object MUST either be defined in the IANA registry or, if for local use only, begin with the substring "temp-". The IANA registry (Section 5.2) will never register names that begin with "temp-".

All names MUST consist only of lower-case ASCII characters, digits, and hyphens (that is, Unicode characters U+0061 through 007A, U+0030 through U+0039, and U+002D), and MUST be 63 characters or shorter. As defined in Section 5.2, the IANA registry will not register names that begin with "temp-", so these names can be used freely by any implementer.

Note that the message returned by the resolver MUST be in I-JSON format. I-JSON requires that the message MUST be encoded in UTF8.

This document only defines one element that can be returned: "inventory". All other elements will be defined in other documents.

4.1. The "inventory" name

The "inventory" name lists all of the types of information for which the resolver has data. The value is an array of strings.

4.2. Example

The I-JSON object that a resolver returns might look like the following:

```
{
  "temp-field2": 42,
  "temp-field1": [ "There is", "no \u000B!" ],
  "inventory": [ "inventory", "temp-field1", "temp-field2" ]
}
```

As specified in [RFC7493], the I-JSON object is encoded as UTF8. This example has no un-escaped non-ASCII characters only because they are not currently allowed in Internet Drafts. For example, the exclamation mark in the second name/value pair could instead be the double exclamation mark character, U+203C.

[RFC7493] explicitly allows the returned objects to be in any order.

5. IANA Considerations

5.1. RESINFO RRtype

This document defines a new DNS RR type, RESINFO, whose value TBD will be allocated by IANA from the "Resource Record (RR) TYPEs" sub-registry of the "Domain Name System (DNS) Parameters" registry:

Type: RESINFO

Value: TBD

Meaning: Information self-published by a resolver as an I-JSON (RFC 7493) object

Reference: This document

5.2. Registry for DNS Resolver Information

IANA will create a new registry titled "DNS Resolver Information" that will contain definitions of the names that can be used with the protocols defined in this document. The registration procedure is by Expert Review and Specification Required, as defined in [RFC8126].

The specification that is required for registration can be either an Internet-Draft or an RFC. The reviewer for this registry is instructed to generally be liberal in what they accept into the registry: as long as the specification that comes with the registration request is reasonably understandable, the registration should be accepted.

The registry has the following fields for each element:

Name: The name to be used in the JSON object. This name MUST NOT begin with "temp-". This name MUST conform to the definition of "string" in I-JSON [RFC7493] message format.

Value type: The type of data to be used in the JSON object.

Specification: The name of the specification for the registered element.

5.3. resolver-info Well-known URI

Before this draft is complete, mail will be sent to wellknown-uri-review@ietf.org in order to be registered in the "Well-Known URIs" registry at IANA. The mail will contain the following:

URI suffix: resolver-info

Change controller: IETF

Specification document(s): This document

Status: permanent

6. Security Considerations

Unless a DNS request for <reverse-ip>.{in-addr,ip6}.arpa/IN/RESINFO, or a subdomain, as described in Section 2 is sent over DNS-over-TLS (DoT) [RFC7858] or DNS-over-HTTPS (DoH) [RFC8484], or unless the <reverse-ip>.{in-addr,ip6}.arpa zone is signed with DNSSEC, the response is susceptible to forgery. Stubs and resolvers SHOULD use normal DNS methods for avoiding forgery such as query ID randomization and source port randomization. A stub resolver will know if it is using DoT or DoH, and if it is using DoT it will know if the communication is authenticated (DoH is always authenticated).

An application that is using an operating system API to send queries for <reverse-ip>.{in-addr,ip6}.arpa/IN/RESINFO or a subdomain will only know if query went over authenticated DoT or DoH if the API

supports returning that authentication information. Currently, no common APIs support that type of response.

7. References

7.1. 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>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/info/rfc7493>>.
- [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>>.
- [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>>.
- [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>>.

7.2. Informative References

- [I-D.ietf-acme-ip] Shoemaker, R., "ACME IP Identifier Validation Extension", draft-ietf-acme-ip-06 (work in progress), May 2019.
- [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>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <<https://www.rfc-editor.org/info/rfc8615>>.

Acknowledgments

The idea of various types of servers publishing information about themselves has been around for decades. However this idea has not been used in the DNS. This document aims to fix this omission.

Erik Kline suggested using "<reverse-ip>.{in-addr,ip6}.arpa" as the domain name to allow for the possibility of DNSSEC-signed responses.

Authors' Addresses

Puneet Sood
Google

Email: puneets@google.com

Roy Arends
ICANN

Email: roy.arends@icann.org

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

DNSOP Working Group
Internet-Draft
Updates: 7873 (if approved)
Intended status: Standards Track
Expires: December 28, 2019

O. Sury
Internet Systems Consortium
W. Toorop
NLnet Labs
D. Eastlake 3rd
Futurewei Technologies
M. Andrews
Internet Systems Consortium
June 26, 2019

Interoperable Domain Name System (DNS) Server Cookies
draft-sury-toorop-dnsop-server-cookies-00

Abstract

DNS cookies, as specified in RFC 7873, are a lightweight DNS transaction security mechanism that provides limited protection to DNS servers and clients against a variety of denial-of-service and amplification, forgery, or cache poisoning attacks by off-path attackers.

This document provides precise directions for creating Server Cookies so that an anycast server set including diverse implementations will interoperate with standard clients.

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 December 28, 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	4
2. Changes to [RFC7873]	4
3. Constructing a Client Cookie	4
4. Constructing a Server Cookie	5
4.1. The Version Sub-Field	5
4.2. The Reserved Sub-Field	5
4.3. The Timestamp Sub-Field	5
4.4. The Hash Sub-Field	6
5. Updating the Server Secret	6
6. Cookie Algorithms	7
7. IANA Considerations	8
8. References	8
8.1. Normative References	8
8.2. Informative References	9
Appendix A. Acknowledgements	9
Appendix B. Test vectors	9
B.1. Learning a new Server Cookie	9
B.2. The same client learning a renewed (fresh) Server Cookie	10
B.3. Another client learning a renewed Server Cookie	11
B.4. IPv6 query with rolled over secret	12
Authors' Addresses	13

1. Introduction

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

interoperable strong cookies so that an anycast server set including diverse implementations can be easily configured to interoperate with standard clients.

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

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

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

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

1.1. Contents of this document

Section Section 2 summarises the changes to [RFC7873].

In Section Section 3 suggestions for constructing a Client Cookie are given.

In Section Section 4 instructions for constructing a Server Cookie are given.

In Section Section 5 instructions on updating Server Secrets are given.

In Section Section 6 the different hash functions usable for DNS Cookie construction are listed. [FNV] and HMAC-SHA-256-64 [RFC6234] are deprecated and [SipHash-2.4] is introduced as a REQUIRED hash function for server side DNS Cookie implementations.

IANA considerations are in Section 7.

Acknowledgements are in Appendix A.

Test vectors are in Appendix B.

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.

- o "IP Address" is used herein as a length independent term covering both IPv4 and IPv6 addresses.

2. Changes to [RFC7873]

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

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

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

3. Constructing a Client Cookie

The Client Cookie 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 [SipHash-2.4]) is used, there's no need to change Client secret often. It is reasonable to change the Client secret only if it has been compromised or after a relatively long period of time such as no longer than a year.

It is RECOMMENDED but not required that the following pseudorandom function be used to construct the Client Cookie:

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

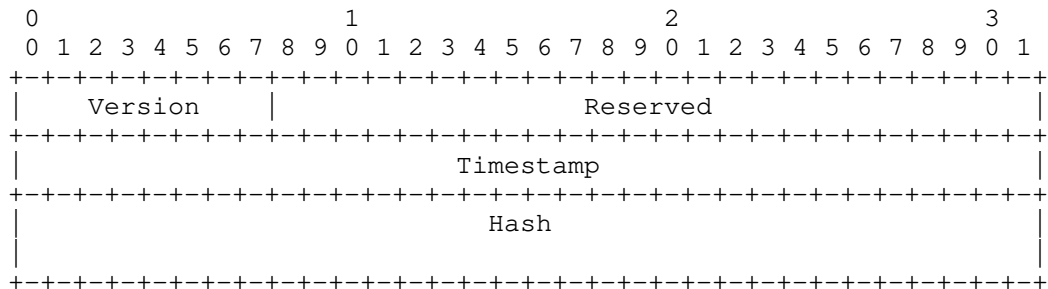
where "|" indicates concatenation.

4. Constructing a Server Cookie

The Server Cookie is effectively a Message Authentication Code (MAC) and should be treated as such. The Server Cookie is calculated from the Client Cookie, a series of Sub-Fields specified below, the Client IP address, and a Server Secret known only to the servers responding on the same address in an anycast set.

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

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



4.1. The Version Sub-Field

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

4.2. The Reserved Sub-Field

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

4.3. The Timestamp Sub-Field

The Timestamp value prevents Replay Attacks and MUST be checked by the server to be within a defined period of time. The DNS Server SHOULD allow Cookies within 1 hour period in the past and 5 minutes

into the future to allow operation of low volume clients and some limited time skew between the DNS servers in the anycast.

The Timestamp value specifies a date and time in the form of a 32-bit unsigned number of seconds elapsed since 1 January 1970 00:00:00 UTC, ignoring leap seconds, in network byte order. All comparisons involving these fields MUST use "Serial number arithmetic", as defined in [RFC1982]

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

4.4. The Hash Sub-Field

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

```
Hash = SipHash2.4(  
    Client Cookie | Version | Reserved | Timestamp | Client-IP,  
    Server Secret )
```

Notice that Client-IP is used for hash generation even though it's not included in the cookie value itself. Client-IP can be either 4 bytes for IPv4 or 16 bytes for IPv6.

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

5. Updating the Server Secret

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

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

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

Stage 1

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

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

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

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

Stage 2

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

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

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

Stage 3

This stage is initiated by the operator when it can be assumed that most clients have learned the new Server Secret.

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

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

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

6. Cookie Algorithms

[SipHash-2.4] is a pseudorandom function suitable as Message Authentication Code. This document REQUIRES compliant DNS Server to use SipHash-2.4 as a mandatory and default algorithm for DNS Cookies to ensure interoperability between the DNS Implementations.

The construction method and pseudorandom function used in calculating and verifying the Server Cookies are determined by the initial

version byte and by the length of the Server Cookie. Additional pseudorandom or construction algorithms for Server Cookies might be added in the future.

7. IANA Considerations

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

Registry Name: DNS Server Cookie Methods

Assignment Policy: Expert Review

Reference: [this document], [RFC7873]

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

Implementation recommendations for Cookie Algorithms [DNSCOOKIE-
IANA]:

Version	Size	Method
0	8-32	reserved
1	8-15	unassigned
1	16	SipHash-2.4 [this document] Section 4
1	17-32	unassigned
2-239	8-32	unassigned
240-254	8-32	private use
255	8-32	reserved

8. References

8.1. Normative References

- [RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <<https://www.rfc-editor.org/info/rfc1982>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7873] Eastlake 3rd, D. and M. Andrews, "Domain Name System (DNS) Cookies", RFC 7873, DOI 10.17487/RFC7873, May 2016, <<https://www.rfc-editor.org/info/rfc7873>>.

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

[SipHash-2.4] Aumasson, J. and D. Bernstein, "SipHash: a fast short-input PRF", 2012, <<https://131002.net/siphash/>>.

8.2. 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>>.

[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>>.

Appendix A. Acknowledgements

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

Appendix B. Test vectors

B.1. Learning a new Server Cookie

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

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


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

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a05862bf552bbd22
;; QUESTION SECTION:
;example.com.                IN      A

;; QUERY SIZE: 52
```

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

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

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

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

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a05862bf552bbd22010000005cf79f11de1bba0952b0ff82 (good)
;; QUESTION SECTION:
;example.com.                IN      A

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

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

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

40 minutes later, the same resolver (client) queries the same server for for "example.org" :


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

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a05862bf552bbd22010000005cf79f11de1bba0952b0ff82
;; QUESTION SECTION:
;example.org.                IN      A

;; QUERY SIZE: 52
```

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

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

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a05862bf552bbd22010000005cf7a87144c909a80f560a7f (good)
;; QUESTION SECTION:
;example.org.                IN      A

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

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

B.3. Another client learning a renewed Server Cookie

Another resolver (client) with IPv4 address 198.51.100.100 sends a request to the same server with a valid Server Cookie that it learned before (at Wed Jun 5 09:46:25 UTC 2019). Note that the Server Cookie has Reserved bytes set, but is still valid with the configured secret; the Hash part is calculated taking along the Reserved bytes.


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

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags;; udp: 4096
; COOKIE: 35fe2405c6e35ce201abcdef5cf78f71f36da1fa58a43879
;; QUESTION SECTION:
;example.com.                IN      A

;; QUERY SIZE: 52
```

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

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

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags;; udp: 4096
; COOKIE: 35fe2405c6e35ce2010000005cf7a9ac70f66429813e466a (good)
;; QUESTION SECTION:
;example.com.                IN      A

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

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

B.4. IPv6 query with rolled over secret

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


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

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: fedba308ab9fddb2010000005cf7c579bbc73b7d5da18dd1
;; QUESTION SECTION:
;example.net.                IN      A

;; QUERY SIZE: 52
```

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

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

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: fedba308ab9fddb2010000005cf7c609449d89428c0e2d92 (good)
;; QUESTION SECTION:
;example.net.                IN      A

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

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

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

Donald E. Eastlake 3rd
Futurewei Technologies
1424 Pro Shop Court
Davenport FL 33896
USA

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

Mark Andrews
Internet Systems Consortium
950 Charter Street
Redwood City CA 94063
USA

Email: marka@isc.org

DNSOP Working Group
Internet-Draft
Updates: 2308, 4033, 4034, 4035 (if
approved)
Intended status: Informational
Expires: January 22, 2020

J. Woodworth
D. Ballew
CenturyLink, Inc.
S. Bindinganaveli Raghavan
Hughes Network Systems
D. Lawrence
Oracle
July 21, 2019

BULK DNS Resource Records
draft-woodworth-bulk-rr-09

Abstract

The BULK DNS resource record type defines a method of pattern-based creation of DNS resource records based on numeric substrings of query names. The intent of BULK is to simplify generic assignments in a memory-efficient way that can be easily shared between the primary and secondary nameservers for a zone.

Ed note

Text inside square brackets ([]) is additional background information, answers to frequently asked questions, general musings, etc. They will be removed before publication. This document is being collaborated on in GitHub at <<https://github.com/ionevez/bulk-rr>>. The most recent version of the document, open issues, etc should all be available here. The authors gratefully accept pull requests.

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 22, 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	3
1.1. Background and Terminology	4
2. The BULK Resource Record	4
2.1. BULK RDATA Wire Format	4
2.2. The BULK RR Presentation Format	6
3. BULK Replacement	7
3.1. Matching the Domain Name Pattern	7
3.2. Record Generation using Replacement Pattern	7
3.2.1. Delimiters	8
3.2.2. Delimiter intervals	8
3.2.3. Padding length	9
3.2.4. Final processing	9
4. Known Limitations	9
4.1. Unsupported Nameservers	10
5. Security Considerations	10
5.1. DNSSEC Signature Strategies	10
5.1.1. On-the-fly Signatures	10
5.1.2. Alternative Signature Scheme	11
5.1.3. Non-DNSSEC Zone Support Only	11
5.2. DDOS Attack Vectors and Mitigation	11
5.3. Implications of Large-Scale DNS Records	11
6. Privacy Considerations	12
7. IANA Considerations	12
8. Acknowledgments	12
9. References	12
9.1. Normative References	12
9.2. Informative References	13
Appendix A. BULK Examples	14
A.1. Example 1	14
A.2. Example 2	14
A.3. Example 3	15

A.4. Example 4	15
A.5. Example 5	15
Authors' Addresses	16

1. Introduction

The BULK DNS resource record defines a pattern-based method for on-the-fly resource record generation. It is essentially an enhanced wildcard mechanism, constraining generated resource record owner names to those that match a pattern of variable numeric substrings. It is also akin to the \$GENERATE master file directive [bind-arm] without being limited to numeric values and without creating all possible records in the zone data.

For example, consider the following record:

```
example.com. 86400 IN BULK A (
    pool-A-[0-255]-[0-255].example.com.
    10.55.${1}.${2}
)
```

It will answer requests for pool-A-0-0.example.com through pool-A-255-255.example.com with the IPv4 addresses 10.55.0.0 through 10.55.255.255.

Much larger record sets can be defined while minimizing the associated requirements for server memory and zone transfer network bandwidth.

This record addresses a number of real-world operational problems that authoritative DNS service providers experience. For example, operators who host many large reverse lookup zones, even for only IPv4 space in in-addr.arpa, would benefit from the disk space, memory size, and zone transfer efficiencies that are gained by encapsulating a simple record-generating algorithm versus enumerating all of the individual records to cover the same space.

Production zones of tens of thousands of pattern-generated records currently exist, that could be reduced to just one BULK RR. These zones can look deceptively small on the primary nameserver and balloon to 100MB or more when expanded,

BULK also allows administrators to more easily deal with singletons, records in the pattern space that are an exception to the normal data generation rules. Whereas a mechanism like \$GENERATE may need to be adjusted to account for these individual records, the processing rules for BULK have explicit records more naturally override the dynamically generated ones. This collision problem is not just a

theoretical concern, but a real source of support calls for providers.

Pattern-generated records are also not only for the reverse DNS space. Forward zones also occasionally have entries that follow patterns that would be well-addressed by the BULK RR.

1.1. Background and Terminology

The reader is assumed to be familiar with the basic DNS and DNSSEC concepts described in [RFC1034], [RFC1035], [RFC4033], [RFC4034], and [RFC4035]; subsequent RFCs that update them in [RFC2181] and [RFC2308]; and DNS terms in [RFC7719].

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] when, and only when, they appear in all capitals, as shown here.

2. The BULK Resource Record

The BULK resource record enables an authoritative nameserver to generate RRs for other types based upon the query received.

The Type value for the BULK RR type is TBD.

The BULK RR is class-independent.

2.1. BULK RDATA Wire Format

The RDATA for a BULK RR is as follows:

```

      1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Match Type           |                               Domain Name Pattern       /
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
/                               /
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
/                               /
/                               /
/                               /
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Match Type identifies the type of the RRset to be generated by this BULK record. It is two octets corresponding to an RR TYPE code as specified in [RFC1035], Section 3.2.1.

Domain Name Pattern consists of a pattern encoded as a wire-format fully qualified domain name. The full name is used so that numeric substrings above the zone cut can be captured in addition to those in the zone. It needs no length indicator for the entire field because the root label marks its end.

Special characters are interpreted as per the following Augmented Backus-Naur Form (ABNF) notation from [RFC5234].

```
match          = 1*(range / string)

range          = "[" [decnum "-" decnum] "]" /
                "<" [hexnum "-" hexnum] ">"
                ; create references for substitution
                ; limit of 32 references
                ; [] is syntactic sugar for 0-255
                ; <> is syntactic sugar for 00-ff

string         = 1*(ctext / quoted-char)

decnum         = 1*decdigit
                ; constrained to 65535 maximum.

hexnum        = 1*hexdigit
                ; constrained to ffff maximum.

octet         = %x00-FF

decdigit      = %x30-39
                ; 0-9

hexdigit      = decdigit / 0x41-0x46 / 0x61-66
                ; 0-9, A-F, a-f

ctext         = <any octet excepting "\">

quoted-char   = "\" octet
                ; to allow special characters as literals
```

Interpretation of the Domain Name Pattern is described in detail in the "BULK Replacement" section. Note that quoted-char must be stored in the wire format to preserve its semantics when the BULK RR is interpreted by nameservers.

The limit of 32 references is meant to simplify implementation details. It is largely but not entirely arbitrary, as it could capture every individual character of the text representation of a full IPv6 address.

Replacement Pattern describes how the answer RRset MUST be generated for the matching query. It needs no length indicator because its end can be derived from the RDATA length minus Match Type and Domain Name Pattern lengths. It uses the following additional ABNF elements:

```

replace      = 1*(reference / string)

reference     = "$" "{" (positions / "*") [options] "}"

positions    = (position / posrange) 0*("," (position / posrange))

posrange     = position "-" position

position     = 1*decnum

options      = delimiter [interval [padding]]

delimiter    = "|" 0*(ctext | quoted-char)
               ; "\" to use "|" as delimiter
               ; "\\\" to use "\" as delimiter

interval     = "|" *2decdigit

padding      = "|" *2decdigit

```

[Is the formatting complexity beyond simple \${1}, \${2}, etc, really worth it? I definitely see how it could make for shorter replacement patterns, but does it enhance their clarity and usability, adding a feature someone really wants?]

The Replacement Pattern MUST end in the root label if it is intended to represent a fully qualified domain name.

2.2. The BULK RR Presentation Format

Match Type is represented as an RR type mnemonic or with [RFC3597]'s generic TYPE mechanism.

Domain Name Pattern is represented as a fully qualified domain name as per [RFC1035] Section 5.1 rules for encoding whitespace and other special characters.

Replacement Pattern is represented by the standard <character-string> text rules for master files as per [RFC1035] section 5.1.

It is suggested that lines longer than 80 characters be wrapped with parenthetical line continuation, per [RFC1035] Section 5.1, starting after Match Type and ending after Replacement Pattern.

3. BULK Replacement

When a BULK-aware authoritative nameserver receives a query for which it does not have a matching name or a covering wildcard, it MUST then look for BULK RRs at the zone apex, selecting all BULK RRs with a Match Type that matches the query type and a Domain Name Pattern that matches the query name. Note that query type ANY will select all Match Types, and all query types match a CNAME or DNAME Match Type. One or more answer RRs will be generated per the replacement rules below. Examples are provided in an appendix.

By only triggering the BULK algorithm when the query name does not exist, administrators are given the flexibility to explicitly override the behaviour of specific names that would otherwise match the BULK record's Domain Name Pattern. This is unlike BIND's \$GENERATE directive, which adds the generated RRs to any existing names.

3.1. Matching the Domain Name Pattern

A query name matches the Domain Name Pattern if the characters that appear outside the numeric ranges match exactly and those within numeric ranges have values that fall within the range. Numeric matches MUST be of the appropriate decimal or hexadecimal type as specified by the delimiters in the pattern. For example, if a range is given as [0-255], then FF does not match even though its value as a hexadecimal number is within the range. Leading zeros in the numeric part(s) of the qname MUST be ignored; for example, 001.example.com, 01.example.com and 1.example.com would all match [].example.com.

When a query name matches a Domain Name Pattern, the value in each numeric range is stored for use by the Replacement Pattern, with reference numbers starting at 1 and counting from the left. For example, matching the query name host-24-156 against host-[0-255]-[0-255] assigns 24 to \${1} and 156 to \${2}.

3.2. Record Generation using Replacement Pattern

The Replacement Pattern generates the record data by replacing the \${...} references with data captured from the query name, and copying all other characters literally.

The simplest form of reference uses only the reference number between the braces, "{" and "}". The value of the reference is simply copied directly from the matching position of the query name.

The next form of reference notation uses the asterisk, "*". With \${*}, all captured values in order of ascending position, delimited by its default delimiter (described below), are placed in the answer. The commercial-at, "@" symbol captures in the same way only in order of descending position.

Numeric range references, such as \${1-4}, replaces all values captured by those references, in order, delimited by the default delimiter described below. To reverse the order in which they are copied, reverse the upper and lower values, such as \${4-1}. This is useful for generating PTR records from query names in which the address is encoded in network order.

Similar to range references, separating positions by commas creates sets for replacement. For example, \${1,4} would be replaced by the first and fourth captured values, delimited its default delimiter. This notation may be combined with the numeric range form, such as \${3,2,1,8-4}.

3.2.1. Delimiters

A reference can specify a delimiter to use by following a vertical bar, "|", with zero or more characters. Zero characters, such as in \${1-3|}, means no delimiter is used, while other characters up to an unescaped vertical bar or closing brace are copied between position values in the replacement. The default delimiter is the hyphen, "-".

3.2.2. Delimiter intervals

A second vertical bar in the reference options introduces a delimiter interval. The default behavior of a multi-position reference is to combine each captured value specified with a delimiter between each. With a delimiter interval the delimiters are only added between every Nth value. For example, \${*|-|4} adds a hyphen between every group of four captured positions. This can be a handy feature in the IPv6 reverse namespace where every nibble is captured as a separate value and generated hostnames include sets of 4 nibbles. An empty or 0 value for the delimiter interval MUST be interpreted as the default value of 1.

3.2.3. Padding length

The fourth and final reference option determines the field width of the copied value. Shorter values MUST be padded with leading zeroes ("0") and longer values MUST be truncated to the width.

The default behavior, and that of an explicit empty padding length, is that the captured query name substring is copied exactly. A width of zero "0" is a signal to "unpad", and any leading zeros MUST be removed. [Unnecessary complexity?]

If a delimiter interval greater than 1 is used, captured values between the intervals will be concatenated and the padding or unpadding applied as a unit and not individually. An example of this would be `${*||4|4}` which would combine each range of 4 captured values and pad or truncate them to a width of 4 characters.

[If this is kept, the element/feature should probably be renamed from "padding" since it is just as likely to truncate.]

3.2.4. Final processing

The string that results from all replacements is converted to the appropriate RDATA format for the record type. If the conversion fails, the SERVFAIL rcode MUST be set on the response, representing a misconfiguration that the server was unable to perform. [The EDNS extended-error code would be useful here.]

The TTL of each RR generated by a BULK RR is the TTL of the corresponding BULK record itself. [BULK should probably have its own TTL field because using that of the record itself feels like bad design. On the other hand, if BULK is never meant to be queried for directly and only appears in authoritative data, its own TTL is pretty useless normally.]

The class for the RRSet is the class of the BULK RR.

If the generated record type is one that uses domain names in its resource record data, such as CNAME, a relative domain names MUST be fully qualified with the origin domain of the BULK RR.

4. Known Limitations

This section defines known limitations of the BULK resource type.

4.1. Unsupported Nameservers

Authoritative nameservers that do not understand the semantics of the new record type will not be able to deliver the intended answers even when the type appears in their zone data. This significantly affects the interoperability of primary versus secondary authorities that are not all running the same software. Adding new RRs which affect handling by authoritative servers, or being unable to add them, is an issue that needs to be explored more thoroughly within dnsop.

5. Security Considerations

Two known security considerations exist for the BULK resource record, DNSSEC and DDOS attack vectors.

5.1. DNSSEC Signature Strategies

DNSSEC was designed to provide validation for DNS resource records, requiring each tuple of owner, class, and type to have its own signature. This essentially defeats the purpose of providing large generated blocks of RRs in a single RR as each generated RR would require its own legitimate RRSIG record.

In the following sections several options are discussed to address this issue. Of the options, on-the-fly provides the most secure solution and NPN [nnp-draft] provides the most flexible.

5.1.1. On-the-fly Signatures

A significant design goal of DNSSEC was to be able to do offline cryptographic signing of zone contents, keeping the key material more secure.

On-the-fly processing requires authoritative nameservers to sign generated records as they are created. Not all authoritative nameserver implementations offer on-the-fly signatures, and even with those that do not all operators will want to keep signing keys online. This solution would either require all implementations to support on-the-fly signing or be ignored by implementations which can not or will not comply.

One possible mitigation for addressing the risk of keeping the zone signing key online would be to continue to keep the key for signing positive answers offline and introduce a second key for online signing of negative answers.

No changes to validating resolvers is required to support this solution.

5.1.2. Alternative Signature Scheme

Previous versions of this draft proposed a new signature scheme using a Numeric Pattern Normalization (NPN) RR. It was a method to support offline signatures for BULK records, with the drawback that is required updates to DNSSEC-aware resolvers.

That mechanism is not specific to BULK and has been removed from the current draft. If there is further interest in pursuing it, it can be reopened as a separate draft.

5.1.3. Non-DNSSEC Zone Support Only

As a final option zones which wish to remain entirely without DNSSEC support may serve such zones without either of the above solutions and records generated based on BULK RRs will require zero support from recursive resolvers.

5.2. DDOS Attack Vectors and Mitigation

As an additional defense against Distributed Denial Of Service (DDOS) attacks against recursive (resolving) nameservers it is highly recommended shorter TTLs be used for BULK RRs than others. While disabling caching with a zero TTL is not recommended, as this would only result in a shift of the attack target, a balance will need to be found. While this document uses 24 hours (86400 seconds) in its examples, values between 300 to 900 seconds are likely more appropriate and is RECOMMENDED. What is ultimately deemed appropriate may differ from zone to zone and administrator to administrator.

[I am unclear how this helps DDOS mitigation against anyone at all, and suspect this section should be removed..]

5.3. Implications of Large-Scale DNS Records

The production of such large-scale records in the wild may have some unintended side-effects. These side-effects could be of concern or add unexpected complications to DNS based security offerings or forensic and anti-spam measures. While outside the scope of this document, implementers of technology relying on DNS resource records for critical decision making must take into consideration how the existence of such a volume of records might impact their technology.

Solutions to the magnitude problem for BULK generated RRs are expected be similar if not identical to that of existing wildcard records, the core difference being the resultant RDATA will be unique for each requested Domain Name within its scope.

The authors of this document are confident that by careful consideration, negative_side-effects produced by implementing the features described in this document can be eliminated from any such service or product.

6. Privacy Considerations

The BULK record does not introduce any new privacy concerns to DNS data.

7. IANA Considerations

IANA is requested to assign numbers for the BULK RR.

8. Acknowledgments

This document was created as an extension to the DNS infrastructure. As such, many people over the years have contributed to its creation and the authors are appreciative to each of them even if not thanked or identified individually.

A special thanks is extended for the kindness, wisdom and technical advice of Robert Whelton (CenturyLink, Inc.) and Gary O'Brien (Secure64 Software Corp).

9. References

9.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.

- [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>>.
- [RFC2317] Eidnes, H., de Groot, G., and P. Vixie, "Classless IN-ADDR.ARPA delegation", BCP 20, RFC 2317, DOI 10.17487/RFC2317, March 1998, <<https://www.rfc-editor.org/info/rfc2317>>.
- [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>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.

9.2. Informative References

- [bind-arm] Internet Systems Consortium, "BIND 9 Configuration Reference", 2016, <<https://ftp.isc.org/isc/bind9/cur/9.9/doc/arm/Bv9ARM.html>>.
- [nnp-draft] Internet Systems Consortium, "Numeric Pattern Normalization (NPN)", 2019, <<https://github.com/ionevez/npn>>.

[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>>.

Appendix A. BULK Examples

A.1. Example 1

```
$ORIGIN 2.10.in-addr.arpa.  
@ 86400 IN BULK PTR (  
    [0-255].[0-255].[0-255].[0-255].in-addr.arpa.  
    pool-${4-1}.example.com.  
)
```

A query received for the PTR of 4.3.2.10.in-addr.arpa will create the references \${1} through \${4} with the first four labels of the query name. The \${4-1} reference in the replacement pattern will then substitute them in reverse with the default delimiter of hyphen between every character and no special field width modifications. The TTL of the BULK RR is used for the generated record, making the response:

```
4.3.2.10.in-addr.arpa 86400 IN PTR pool-10-2-3-4.example.com.
```

A.2. Example 2

```
$ORIGIN 2.10.in-addr.arpa.  
@ 86400 IN BULK PTR (  
    [0-255].[0-255].[0-255].[0-255].in-addr.arpa.  
    pool-${2,1||3}.example.com.  
)
```

Example 2 is similar to Example 1, except that it modifies the replacement pattern. The empty option after the first vertical bar causes no delimiters to be inserted, while the second empty option that would keep the delimiter interval as 1. The latter is relevant because the final value, padding of 3, is applied over each delimiter interval even when no delimiter is used. Not all captures from the substring are required to be used in the response.

The result is that a query for the PTR of 4.3.2.10.in-addr.arpa generates this response:

```
4.3.2.10.in-addr.arpa 86400 IN PTR pool-003004.example.com.
```

[Admittedly you can't do this very effectively without the field width complexity. Is this sort of name common? Does it need

support? Admittedly \$GENERATE had the feature, but is that reason enough?]

[Change this to a hex matching example?]

A.3. Example 3

```
$ORIGIN 0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.
@ 86400 IN BULK PTR (
    <>.<>.<>.<>.<>.<>.<>.<>.<>.<>.<>.<>.<>.<>.<>.<>
    poolAA-#{16-8|-|4}.example.com.
)
```

This example introduces IPv6 where 16 individual nibbles are captured and the last 8 are combined into 2 blocks of 4, separated by a hyphen.

A query for the IP of 2001:db8::dead:beef results in a PTR RR with the value of poolAA-dead-beef.example.com.

A.4. Example 4

```
$ORIGIN example.com.
@ 86400 IN BULK AAAA (
    poolAA-<0-ffff>-<0-ffff>.example.com.
    ${@|.|1}.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.
)
```

This example performs the reverse of example 3, where a query of poolAA-dead-beef.example.com captures "dead" and "beef", reversing the nibbles and using a dot (.) as the delimiter to form a valid AAAA record.

A.5. Example 5

This example contains a classless IPv4 delegation on the /22 CIDR boundary as defined by [RFC2317]. The network for this example is "10.2.0/22" delegated to a nameserver "ns1.sub.example.com.". RRs for this example are defined as:

```
$ORIGIN 2.10.in-addr.arpa.
@ 7200 IN BULK CNAME [0-255].[0-3] ${*|.}.0-3
0-3 86400 IN NS ns1.sub.example.com.
```

A query for the PTR of 25.2.2.10.in-addr.arpa is received and the BULK record with the CNAME Match Type matches all query types. 25 and 2 are captured as references, and joined in the answer by the period (".") character as a delimiter, with ".0-3" then appended

literally and fully qualified by the origin domain. The final synthesized record is:

```
25.2.2.10.in-addr.arpa 7200 IN CNAME 25.2.0-3.2.10.in-addr.arpa.
```

[Without \$* and options complexity, the pattern to get the same result is just \${1}.\${2}.0-3 which is not really significantly onerous to enter, and slightly less arcane looking to comprehend.]

Authors' Addresses

John Woodworth
CenturyLink, Inc.
4250 N Fairfax Dr
Arlington VA 22203
USA

Email: John.Woodworth@CenturyLink.com

Dean Ballew
CenturyLink, Inc.
2355 Dulles Corner Blvd, Ste 200 300
Herndon VA 20171
USA

Email: Dean.Ballew@CenturyLink.com

Shashwath Bindinganaveli Raghavan
Hughes Network Systems
11717 Exploration Lane
Germantown MD 20876
USA

Email: shashwath.bindinganaveliraghavan@hughes.com

David C Lawrence
Oracle

Email: tale@dd.org