Abstract

This document outlines a mechanism by which a registered domain can publicly document a relationship with a different registered 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 September 7, 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.
1. Introduction

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

Determining relationships between registered 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 because it is not clear that the two domains are in fact related. This specification provides a way for one domain to explicitly document a relationship with another, utilizing DNS records.

Possible use cases include:
where a company has websites in different languages, and would like to correlate their ownership more easily, consider "example.de" and "example.ie" registered by regional offices of the same company;

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;

when doing Internet surveys, we should be able to provide more accurate results if we have information as to which domains are related.

It is not a goal of this specification to provide a high-level of assurance that two domains are definitely related, nor to provide fine-grained detail about the kind of relationship that may exist between domains.

Using "Related Domains By DNS", or "RDBD", it is possible to declare that two domains are related.

We include 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., "example.com") and a reference from the related-domain (e.g., "dept-example.com") contains a signature (verifiable with the "example.com" public key) over the text representation (‘A-label’) of the two domain names (plus a couple of other inputs).

RDBD is intended to demonstrate a relationship between registered domains, not individual hostnames. That is to say that the relationship should exist between "example.com" and "dept-example.com", not "foo.example.com" and "bar.dept-example.com" (where those latter two are hosts).

There already exists Vouch By Reference (VBR) [RFC5518], however this only applies to email. RDBD could be a more general purpose solution that could be applied to other use cases, as well as for SMTP transactions.

This document describes the various options, how to create records, and the method of validation, if the option to use digital signatures is chosen.
1.1. 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:

- Relating-domain: this refers to the domain that is declaring a relationship exists. (This was called the "parent/primary" in -00).
- 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 two new RRTYPES, an optional one for the relating-domain (RDBDKEY) to store a public key for when signatures are in use and one for use in related-domains (RDBD).

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.

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? I’ve no idea really:-)])]
2.2. RDBD Resource Record Definition

The RDBD resource record is published at the apex of the related-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.

The RDBD RR is class independent.

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

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

```
+-----------------------------+
<table>
<thead>
<tr>
<th>rdbd-tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>relating-domain name</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>key-tag</td>
</tr>
<tr>
<td>sig-alg</td>
</tr>
<tr>
<td>signature</td>
</tr>
</tbody>
</table>
+-----------------------------+
```

The rdbd-tag field MUST contain the value zero. Later specifications can define new rdbd-tag values.

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.

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 an empty string. [[Is that the right way to have optional fields in RRs? Not sure.]]

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>
related=<related-domain>
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.

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. Directionality and Cardinality

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

If one domain has relationships with many others, then the relevant
RDBD RRs (and RDBDKEY RRs) can be published to represent those.

4. Required Signature Algorithms

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.

Implementations producing RDBD RRs SHOULD support optional signing of
those and production of RDBDKEY RRs.

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]

RSA keys SHOULD use a 2048 bit or longer modulus.
Implementations of this specification that support signing or verifying signatures SHOULD support use of Ed25519 (sig-alg==15).

[ RFC8080 ] [ RFC8032 ]

5. Validation

A validated signature is solely meant to be additional evidence that the two domains are related. The existence of this relationship is not meant to state that the data from either domain should be considered as more trustworthy.

6. Security Considerations

6.1. Efficacy 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.

If the RDBDKEY value has been cached, or is otherwise known via some sufficiently secure mechanism, then the RDBD signature does confirm that the holder of the private key (presumably the relating-domain) considered that the relationship with the related-domain was real at some point in time.

6.2. 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.

If the relating-domain has DNSSEC deployed, but the related-domain does not, then the optional signature can (in a sense) extend the DNSSEC chain to cover the RDBD RR in the related-domain’s zone.

If both domains have DNSSEC deployed, and if the relating-domain public key has been cached, then the the signature mechanism provides additional protection against active attacks involving a parent of one of the domains. Such attacks may in any case be less likely and detectable in many scenarios as they would be generic attacks against DNSSEC-signing (e.g. if a registry injected a bogus DS for a
relating-domain into the registry’s signed zone). If the public key from the relevant RDNDKEY RRs is read from the DNS at the same time as a related RDBD RR, then the signature mechanism provided here may provide little additional value over and above DNSSEC.

6.3. Lookup Loops

It’s conceivable that an attacker could create a loop of relationships, such as a.com->b.com->c.com->a.com or similar. This could cause a resource issue for any automated system. A system SHOULD only perform three lookups from the first domain (a.com->b.com->c.com->d.com). The related and relating-domains 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.

7. 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.]]

8. 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, Andrew Sullivan, Suzanne Woolf, 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.

9. Informative References


Appendix A. Examples

[[TODO: script up generation of all samples – it’s not unlikely we mucked up somewhere below when generating ‘em partly-manually;-)]]

A.1. Sample Unsigned RDBD RR

When example.com is the relating-domain and dept-example.com is the related-domain, an unsigned RDBD RR would look like this in a zone file:

    dept-example.com. IN 3600 RDBD 0 example.com.

The following is equivalent to the above:

    dept-example.com. IN 3600 RDBD 0 example.com. 0 0 ""
A.2. Sample RSA Signature

Appendix C of [RFC6376] has some reference material on how to create a set of keys for use in this type of use case. The RSA key length is recommended to be at least 2048 bits instead of the 1024 recommended in that appendix.

Creation of keys:

$ openssl genrsa -out rsa.private 2048
$ openssl rsa -in rsa.private -out rsa.public -pubout -outform PEM

Sample Key:

```
rsa.private:
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA2LNjBAdNatZOMdd3hlemZ8a0onOce05g1WKnKzrYDCfH4LZ
kXOPzALJvz4yKMHW5ykOz90G1LM18ns8Ly9ztBxc4obY5wnQp14nbvOdf6vyLy
7Gqgg+pOj6Yrcs7YjDliyYapHwYuLkMrsL6MDLU95S1qskzZLPVqwT80xch
U65HlpKrz2luSAySzyNeF58pRea3DspBkLy5hCDhr2+6GF2q91J9qMopd2FZX
Hkvz13TFTx6Gjp5LTab2dy3tED7vbF/EyQfVwrs4495a80UkOBv7YV4YkgKbFySSK
GPMnWoPBv7hCQEaurWLM97EU0u3U1WqTj1QIDAQABfdNaIB/Aeagwrq6w4/0X
Bkg41Q9g6vnWqWcW5Z40rQg+MnsnshKpVR+krIGU/ft7/vaIzIFPTGr7VWX13
+o2g/1sRFPYUIItjalujqjxehVWHH1saYcbZ21A1vix9QtkgjvBvF6GqZqfl1MJfrc32
QP36a/iAivjdHHN67Bdkgr6VEV1S52PmW4aLjHCGsYDUM4zRc14exzw+rst1
z2seOhnJmYdc+VnKeg5KlENlx2Z3zoY3je/OsfNjTkjAPRPkpiqve3hu3eDP
Pok27BMM+oxk29Fx6AgC99eDr9batTa/a8q7NYmkVRLq/Jd0FXUoDDNd393Ae4n
54qqucECQYE/Xuct5G3l5i055LVxdwB+1wG1JNe11B+O1t6B8w49po
vk/fFVHMEV2BoRr4EB8Q+o1CRBNtzTQUXQynMbxZL2N2+X3FrkD5SGCQy7GzD
wFdpY3cENshoulBnt4/hPWL352XMK3ygBJeGhbTuMVdWwrXTNo2wUCgYEA2tepE
+bg91YUjAg/CEpDwn+8ZxhRnBDzln8Grli+arSwuMW8O9yGpealbwywnxB
vUiskE43Ccgsth8KRY8wDB2AqOnRvEsVkJOK8w/ONSX1WTPcF78xmmNSvObS4Rv
quMc6HTMaetCM/10pddCY3/rls9FTEsF36RXpECgYAE5AF6mHyWb4AT3/ERMtss
ZAwu7FSfx8+Vl2ZUItvR7VH104XTE7MO5plb2796rXKXWlq2G72rznA+5JXjdWW
FW4cOFds/AY7VpIXQ6wr3Ctcet9GWEAe7fCJRZnyH8K7Ejgn8BuFtmYfTTzrWOUP
bksHriRdXJjVxJ1U8hyYekcGyBhM9i24HTVTVnUtyTnIb+ol1jsjxWAL7c6u4O
gcCGu2w6CLieNXXzRbZ6Mik4OJCqpsTuNczsStymoObWas8nyvReason1eSnAzd
dXOGx0hWPSasNswEdVvMAYqLybN6pg+78quAQ4AW+zqoGzjDmpJpSArunJiy2yQ
G7MNQQKbGZkteCEGu2zrx8gyVT586es7P1hp2j8Wabxhd+DMNUEB7c4H2zd85X
AXJxNj2VQWLoL0s10yP9g9lWTVCeZ3MqgQsN1QamN9KjxA46I1tpWzq3Nw2Tk1
m7RB9F9m9mn19/azK7Yuij11/O3cNULIEWcraKqydPfvmNyEtP
-----END RSA PRIVATE KEY-----
```
rsapublic:
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2LNjBA2HAtZOMdd3hlemZ
F8a0onOeEo5g1KWnKzryDCfH4LZkX0PzAJvz4yKMHW5ykOz9OzGL01GM18ns8Ly
9ztBXc4obY5wnQpI4nbvOdf6vyLy7Gqpp+dj6RrYcSYdJdLitiYapHwRyuKmERLQL
6MDWLU9ZSWlqskzLVPgwqtT80xchU65HipKkr21u5AYsZyyNEf58pRea3D3pBkLy
5hCDhr2+6GF2q9I9qMopd2P/ZXxHkvz13TFTX6GjP5LTsb2dy3tED7vbf/EyQfV
wrs4495a8OUk0By7V4YkgKBYYSSkGPMhWoPbV7hCQjEaURWLM9J7EUou3U1W1qTj
1QIDAQAB
-----END PUBLIC KEY-----

To calculate the key-tag as specified in Appendix B of [RFC4034] we
used python code from: <https://www.v13.gr/blog/?p=239>

File containing to-be-signed data:

```
$ cat to-be-signed-8.txt
relating=example.com
related=foo-example.com
rdbd-tag=0
key-tag=65498
sig-alg=8
$
```

To sign that file:

```
$ od -x to-be-signed-8.txt
00000000 6572 616c 6974 676e 653d 6178 706d 656c
0000020 632e 6d6f 720a 6c65 7461 6465 643d 7065
0000040 2d74 7865 6d61 6c70 2e65 6f63 0a6d 6472
0000060 6462 742d 6761 303d 6b0a 7965 742d 6761
0000100 363d 3435 3839 730a 6769 612d 676c 383d
0000120 000a
0000121
```

$ openssl dgst -sha256 -sign rsa.private \
   -out rsa.sig to-be-signed-8.txt
$ od -x rsa.sig
 0000000 087c d5c9 375f dcba 9edf ce25 e353 9fb9
 0000020 6ef4 ca9f a167 6d91 71bb 7487 5edd fe30
 0000040 452e d104 724f f593 009b be3f 6006 ba77
 0000060 c1f5 edc6 e207 7ab0 69a1 79bf 18e6 eea3
 0000100 3562 6ca4 dc73 22c3 1e35 d15c 44be f63
 0000120 ac68 f61e ea34 432d 9e12 2325 d48c 2fd9
 0000140 330d 1caf 5761 6714 eed2 c7e2 47f1 2cla
 0000160 c35b e45e 833b e343 a8e2 3dbf 1a73 02a8
 0000200 c686 7240 aa69 d68a 0868 8e3e 2a0d 057
 0000220 32df 0e62 4679 3f4e 8af8 0716 1a6d 4300
 0000240 ac68 f61e ea34 432d 9e12 2325 d48c 2fd9
 0000260 ab06 3d49 bb42 a84a 071a b959 2d27 3eea
 0000300 c9de 0781 dc5b 22c3 1e35 d15c 44be f63
 0000320 2fbe adee f521 3b75 9c67 66a8 d217 4fd6
 0000340 90da 9423 9db8 f63 7110 1436 f70e 80a2
 0000360 3a8c 25f1 3655 44a2 a585 d87d ca99 aac9
 0000400

The presentation form of a signed RDBD record (with a 3600 TTL) would be:

department-example.com. 3600 RDBD 0 example.com. 65498 8 (hfnhVS17ZMF1tG2qU+4vyCPbfSMutxuV8zEyBv7Gsh0cKMOW
VLFBK116wRUb7wVgG9T8nXyUCkJdQtidEJWftwVZ8Xbzo
tJPMq9HbV2aAnfmx4HxxAMHCpX9QJc20K/5V0bdZm2eZnX14
jd9JsLGuez2wVe1Ckw0x6z/tA61SHmDIFJB5zeKbubvN14y
ABaNE88pxoj7EMVQD/nVoag2MqtasiaMS3kbvkYXC3g25hngM
mZH+kRXGi8ePeyL/ai805n3X2bksrffPu1LQsVC03UEm9Vn
YJgS7jsnvNXwVJpJ9zyWmVbgoR3/5vU2zPvKjSgL7K1n
fPNpg== )

The base64 encoded value for the signature can be produced using:

$ base64 -w48 rsa.sig
hfnhVS17ZMF1tG2qU+4vyCPbfSMutxuV8zEyBv7Gsh0cKMOW
VLFBK116wRUb7wVgG9T8nXyUCkJdQtidEJWftwVZ8Xbzo
tJPMq9HbV2aAnfmx4HxxAMHCpX9QJc20K/5V0bdZm2eZnX14
jd9JsLGuez2wVe1Ckw0x6z/tA61SHmDIFJB5zeKbubvN14y
ABaNE88pxoj7EMVQD/nVoag2MqtasiaMS3kbvkYXC3g25hngM
mZH+kRXGi8ePeyL/ai805n3X2bksrffPu1LQsVC03UEm9Vn
YJgS7jsnvNXwVJpJ9zyWmVbgoR3/5vU2zPvKjSgL7K1n
fPNpg==

Brotman & Farrell Expires September 7, 2019 [Page 12]
To verify, with "rsa.sig" containing the above signature:

```
$ openssl dgst -sha256 -verify rsa.public \
   -signature rsa.sig to-be-signed.txt
Verified OK
```

The RDBDKEY RR for this example would be:

```text
example.com. 3600 RDBDKEY 0 3 8 (LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KUTU1JQklqQU5CZ2txaGtpRz1lMEJBUUVGQUFPQ0ROEFNSU1CQ2dLQ0FRRUEyTE5qQkFkTkF0Wk9NZGQzaGxlOqaRjhhMG9uT2NFbzVnMUtXbkt6cn1EQZ2ZINExaalhPUHpbSznE6NHLTUhXNX1rT3o5T3pHTDAxR01sOG5zOE55cj16fJYYzRvVl1kd25RccGwObmJ2T2RmNnZ5THk3R3FncTkaJ2Scn1JU1KZExpdG1ZXYB1dJ5dUttRvJsUUwKKn1EV0xVOVpTV2xc2t6TF2Q23xdFQ4MHhjaP2NUhpEtrcjiJsdVNBeVNaexK0RNY1OHSZWEzRDNwQm8tMeQo1aENeAIyKzZHRjJxOWxxK0XFnb3BkM1AvWlh4SGt2emwzVEZ0WDZHa1A1FTRzYjJkeTN0RQ3dmJmL05WZWdncyxzQ0OTVhOE9va09CeTdfWNFlzO0t1R11TU2tHUG1oV29QY1Y3aENRakVBBVJXTE05SjdFVW91M1UxVOxlxVOGKMFJREFRQUIKLS0tLS1F
TkQgUFVCTE1oDIEtFWS0tLS0tCgo= )
```

A.3. Sample Ed25519 Signature

Since OpenSSL does not yet support Ed25519 singing via its command line tool, we generate our example using the python script below. This uses the python library from Appendix A of [RFC8032].
#!/usr/bin/env python3
# CODE_BEGINS
import sys, binascii
from eddsa2 import Ed25519

# secret chosen to be 32 octets funny enough :-)  
secret="rdbd-example0001rdbd-example0002".encode('utf-8')  
privkey,pubkey = Ed25519.keygen(secret)  
msg=open('to-be-signed-15.txt','r').read().encode('utf-8')  
signature = Ed25519.sign(privkey, pubkey, msg)

print("private:"+ str(binascii.hexlify(privkey)))  
print("public:"+ str(binascii.hexlify(pubkey)))  
print("sig:"+ str(binascii.hexlify(signature)))  
print("to-be-signed:" + str(msg))

with open("ed25519.sig", "wb") as sigf:  
    sigf.write(signature)  
with open("ed25519.pub","wb") as pubf:  
    pubf.write(pubkey)

# CODE_ENDS

The to-be-signed-15.txt file contains:

$ cat to-be-signed-15.txt
relating=example.com  
related=dept-example.com  
rdbd-tag=0  
key-tag=35988  
sig-alg=15  
$

The output when the above code is run (with some spacing added) is:


The presentation form for an RDBD RR would then be:

department.example.com. 3600 RDBD 0 example.com. 35988 15 (RmqAzmN3seS+xWPYW41VvUpRpbkcHB5GqcTkhZVfDjoXMyKwF5gRtAGbCxSs6F0ILWa+WJ4KwxL1q1XhG+Bw=)

The RDBDKKEY for this example would be:

e.example.com. 3600 RDBDKKEY 0 3 15 (NT/DHhFoyR8K91lsJv1E7ff1nGhOnRr s+yGvo01tkg=)

Appendix B. Changes and Open Issues

[[RFC editor: please delete this appendix]]

B.1. Changes from -00 to -01

- Changed from primary/secondary to relating/related (better suggestions are still welcome)
- Moved away from abuse of TXT RRs
- 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

B.2. Open Issues

Current open github issues include:

o #5: specify input for signing more precisely - e.g. is there a CR or NULL or not

o #6: what, if anything, does rdbd for example.com mean for foo.example.com?

These can be seen at: <https://github.com/abrotman/related-domains-by-dns/issues>

Authors’ Addresses

Alex Brotman
Comcast, Inc

Email: alex_brotman@comcast.com

Stephen Farrell
Trinity College Dublin

Email: stephen.farrell@cs.tcd.ie
Abstract

Some DNS recursive resolvers have longer-than-desired round-trip times to the closest DNS root server. Some DNS recursive resolver operators want to prevent snooping of requests sent to DNS root servers by third parties. Such resolvers can greatly decrease the round-trip time and prevent observation of requests by running a copy of the full root zone on the same server, such as on a loopback address. This document shows how to start and maintain such a copy of the root zone that does not pose a threat to other users of the DNS, at the cost of adding some operational fragility for the operator.

This draft will update RFC 7706. See Section 1.1 for a list of topics that will be added in the update.

[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/wkumari/draft-kh-dnsop-7706bis. The most recent version of the document, open issues, and so on should all be available there. The authors gratefully accept pull requests.]
DNS recursive resolvers have to provide answers to all queries from their customers, even those for domain names that do not exist. For each queried name that has a top-level domain (TLD) that is not in
the recursive resolver’s cache, the resolver must send a query to a root server to get the information for that TLD, or to find out that the TLD does not exist. Research shows that the vast majority of queries going to the root are for names that do not exist in the root zone because negative answers are sometimes cached for a much shorter period of time.

Many of the queries from recursive resolvers to root servers get answers that are referrals to other servers. Malicious third parties might be able to observe that traffic on the network between the recursive resolver and root servers.

The primary goals of this design are to provide more reliable answers for queries to the root zone during network attacks, and to prevent queries and responses from being visible on the network. This design will probably have little effect on getting faster responses to stub resolver for good queries on TLDs, because the TTL for most TLDs is usually long-lived (on the order of a day or two) and is thus usually already in the cache of the recursive resolver; the same is true for the TTL for negative answers from the root servers. (Although the primary goal of the design is for serving the root zone, the method can be used for any zone.)

This document describes a method for the operator of a recursive resolver to have a complete root zone locally, and to hide these queries from outsiders. The basic idea is to create an up-to-date root zone server on the same host as the recursive server, and use that server when the recursive resolver looks up root information. The recursive resolver validates all responses from the root server on the same host, just as it would all responses from a remote root server.

This design explicitly only allows the new root zone server to be run on the same server as the recursive resolver, in order to prevent the server from serving authoritative answers to any other system. Specifically, the root server on the local system MUST be configured to only answer queries from the resolvers on the same host, and MUST NOT answer queries from any other resolver.

At the time that RFC 7706 was published, it was considered controversial: there was not consensus on whether this was a "best practice". In fact, many people felt that it is an excessively risky practice because it introduced a new operational piece to local DNS operations where there was not one before. Since then, the DNS operational community has largely shifted to believing that local serving of the root zone for an individual resolver is a reasonable practice. The advantages listed above do not come free: if this new system does not work correctly, users can get bad data, or the entire
recursive resolution system might fail in ways that are hard to diagnose.

This design uses authoritative name server software running on the same machine as the recursive resolver. Thus, recursive resolver software such as BIND or modern versions of common open source recursive resolver software do not need to add new functionality, but other recursive resolver software might need to be able to talk to an authoritative server running on the same host.

A different approach to solving some of the problems discussed in this document is described in [RFC8198].

1.1. Updates from RFC 7706

RFC 7706 explicitly required that the root server instance be run on the loopback interface of the host running the validating resolver. However, RFC 7706 also had examples of how to set up common software that did not use the loopback interface. Thus, this document loosens the restriction on the interface but keeps the requirement that only systems running on that single host be able to query that root server instance.

Removed the prohibition on distribution of recursive DNS servers including configurations for this design because some already do, and others have expressed an interest in doing so.

Added the idea that a recursive resolver using this design might switch to using the normal (remote) root servers if the local root server fails.

Refreshed the list of where one can get copies of the root zone.

Added examples of other resolvers and updated the existing examples.

[ This section will list all the changes from RFC 7706. For this draft, it is also the list of changes that we will make in future versions of the daft. ]

[ Make the use cases explicit. Be clearer that a real use case is folks who are worried that root server unavailability due to DDoS against them is a reason some people would use the mechanisms here. ]

[ Describe how slaving the root zone from root zone servers does not fully remove the reliance on the root servers being available. ]

[ Other new topics might go here. ]
1.2. Requirements Notation

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

2. Requirements

In order to implement the mechanism described in this document:

- The system MUST be able to validate a zone with DNSSEC [RFC4033].
- The system MUST have an up-to-date copy of the key used to sign the DNS root.
- The system MUST be able to retrieve a copy of the entire root zone (including all DNSSEC-related records).
- The system MUST be able to run an authoritative server for the root zone on the same host. The root server instance MUST only respond to queries from the same host. One way to assure not responding to queries from other hosts is to make the address of the authoritative server one of the loopback addresses (that is, an address in the range 127/8 for IPv4 or ::1 in IPv6).

A corollary of the above list is that authoritative data in the root zone used on the local authoritative server MUST be identical to the same data in the root zone for the DNS. It is possible to change the unsigned data (the glue records) in the copy of the root zone, but such changes could cause problems for the recursive server that accesses the local root zone, and therefore any changes to the glue records SHOULD NOT be made.

3. Operation of the Root Zone on the Local Server

The operation of an authoritative server for the root in the system described here can be done separately from the operation of the recursive resolver, or it might be part of the configuration of the recursive resolver system.

The steps to set up the root zone are:

1. Retrieve a copy of the root zone. (See Appendix A for some current locations of sources.)

2. Start the authoritative server with the root zone on an address on the host that is not in use. For IPv4, this could be 127.0.0.1, but if that address is in use, any address in 127/8 is
acceptable. For IPv6, this would be ::1. It can also be a publicly-visible address on the host, but only if the authoritative server software allows restricting the addresses that can access the authoritative server, and the software is configured to only allow access from addresses on this single host.

The contents of the root zone MUST be refreshed using the timers from the SOA record in the root zone, as described in [RFC1035]. This inherently means that the contents of the local root zone will likely be a little behind those of the global root servers because those servers are updated when triggered by NOTIFY messages.

If the contents of the root zone cannot be refreshed before the expire time in the SOA, the local root server MUST return a SERVFAIL error response for all queries sent to it until the zone can be successfully be set up again. Because this would cause a recursive resolver on the same host that is relying on this root server to also fail, a resolver might be configured to immediately switch to using other (non-local) root servers if the resolver receives a SERVFAIL response from a local root server.

In the event that refreshing the contents of the root zone fails, the results can be disastrous. For example, sometimes all the NS records for a TLD are changed in a short period of time (such as 2 days); if the refreshing of the local root zone is broken during that time, the recursive resolver will have bad data for the entire TLD zone.

An administrator using the procedure in this document SHOULD have an automated method to check that the contents of the local root zone are being refreshed; this might be part of the resolver software. One way to do this is to have a separate process that periodically checks the SOA of the root zone from the local root zone and makes sure that it is changing. At the time that this document is published, the SOA for the root zone is the digital representation of the current date with a two-digit counter appended, and the SOA is changed every day even if the contents of the root zone are unchanged. For example, the SOA of the root zone on January 2, 2018 was 2018010201. A process can use this fact to create a check for the contents of the local root zone (using a program not specified in this document).

4. Using the Root Zone Server on the Same Host

A recursive resolver that wants to use a root zone server operating as described in Section 3 simply specifies the local address as the place to look when it is looking for information from the root. All responses from the root server MUST be validated using DNSSEC.
Note that using this simplistic configuration will cause the recursive resolver to fail if the local root zone server fails. A more robust configuration would cause the resolver to start using the normal remote root servers when the local root server fails (such as if it does not respond or gives SERVFAIL responses).

See Appendix B for more discussion of this for specific software.

To test the proper operation of the recursive resolver with the local root server, use a DNS client to send a query for the SOA of the root to the recursive server. Make sure the response that comes back has the AA bit in the message header set to 0.

5. Security Considerations

A system that does not follow the DNSSEC-related requirements given in Section 2 can be fooled into giving bad responses in the same way as any recursive resolver that does not do DNSSEC validation on responses from a remote root server. Anyone deploying the method described in this document should be familiar with the operational benefits and costs of deploying DNSSEC [RFC4033].

As stated in Section 1, this design explicitly only allows the new root zone server to be run on the same host, answering queries only from resolvers on that host, in order to prevent the server from serving authoritative answers to any system other than the recursive resolver. This has the security property of limiting damage to any other system that might try to rely on an altered copy of the root.

6. References

6.1. Normative References


6.2. Informative References

[Manning2013]  Manning, W., "Client Based Naming", 2013,

              DNSSEC-Validated Cache", RFC 8198, DOI 10.17487/RFC8198,

Appendix A. Current Sources of the Root Zone

The root zone can be retrieved from anywhere as long as it comes with
all the DNSSEC records needed for validation. Currently, one can get
the root zone from ICANN by zone transfer (AXFR) over TCP from DNS
servers at xfr.lax.dns.icann.org and xfr.cjr.dns.icann.org.

Currently, the root can also be retrieved by AXFR over TCP from the
following root server operators:

- b.root-servers.net
- c.root-servers.net
- d.root-servers.net
- f.root-servers.net
- g.root-servers.net
- k.root-servers.net

It is crucial to note that none of the above services are guaranteed
to be available. It is possible that ICANN or some of the root
server operators will turn off the AXFR capability on the servers
listed above. Using AXFR over TCP to addresses that are likely to be
anycast (as the ones above are) may conceivably have transfer
problems due to anycast, but current practice shows that to be
unlikely.

To repeat the requirement from earlier in this document: if the
contents of the zone cannot be refreshed before the expire time, the
server MUST return a SERVFAIL error response for all queries until
the zone can be successfully be set up again.
Appendix B. Example Configurations of Common Implementations

This section shows fragments of configurations for some popular recursive server software that is believed to correctly implement the requirements given in this document. The examples have been updated since the publication of RFC 7706.

The IPv4 and IPv6 addresses in this section were checked recently by testing for AXFR over TCP from each address for the known single-letter names in the root-servers.net zone.


BIND 9.12 acts both as a recursive resolver and an authoritative server. Because of this, there is "fate-sharing" between the two servers in the following configuration. That is, if the root server dies, it is likely that all of BIND is dead.

Note that a future version of BIND will support a much more robust method for creating a local mirror of the root or other zones; see Appendix B.3.

Using this configuration, queries for information in the root zone are returned with the AA bit not set.

When slaving a zone, BIND 9.12 will treat zone data differently if the zone is slaved into a separate view (or a separate instance of the software) versus slaved into the same view or instance that is also performing the recursion.

Validation: When using separate views or separate instances, the DS records in the slaved zone will be validated as the zone data is accessed by the recursive server. When using the same view, this validation does not occur for the slaved zone.

Caching: When using separate views or instances, the recursive server will cache all of the queries for the slaved zone, just as it would using the traditional "root hints" method. Thus, as the zone in the other view or instance is refreshed or updated, changed information will not appear in the recursive server until the TTL of the old record times out. Currently, the TTL for DS and delegation NS records is two days. When using the same view, all zone data in the recursive server will be updated as soon as it receives its copy of the zone.
view root {
    match-destinations { 127.12.12.12; };
    zone "." {
        type slave;
        file "rootzone.db";
        notify no;
        masters {
            199.9.14.201; # b.root-servers.net
            192.33.4.12; # c.root-servers.net
            199.7.91.13; # d.root-servers.net
            192.5.5.241; # f.root-servers.net
            192.112.36.4; # g.root-servers.net
            193.0.14.129; # k.root-servers.net
            192.0.47.132; # xfr.cjr.dns.icann.org
            192.0.32.132; # xfr.lax.dns.icann.org
            2001:500:200::b; # b.root-servers.net
            2001:500:2::c; # c.root-servers.net
            2001:500:2d::d; # d.root-servers.net
            2001:500:2f::f; # f.root-servers.net
            2001:500:12::d0d; # g.root-servers.net
            2001:7fd::1; # k.root-servers.net
            2620:0:2830:202::132; # xfr.cjr.dns.icann.org
            2620:0:2d0:202::132; # xfr.lax.dns.icann.org
        };
    };
};

view recursive {
    dnssec-validation auto;
    allow-recursion { any; };
    recursion yes;
    zone "." {
        type static-stub;
        server-addresses { 127.12.12.12; };
    };
};

B.2. Example Configuration: Unbound 1.8

Similar to BIND, Unbound starting with version 1.8 can act both as a recursive resolver and an authoritative server.
auth-zone:
  name: ".
master: 199.9.14.201         # b.root-servers.net
master: 192.33.4.12          # c.root-servers.net
master: 199.7.91.13          # d.root-servers.net
master: 192.5.5.241          # f.root-servers.net
master: 192.112.36.4         # g.root-servers.net
master: 193.0.14.129         # k.root-servers.net
master: 192.0.47.132         # xfr.cjr.dns.icann.org
master: 192.0.32.132         # xfr.lax.dns.icann.org
master: 2001:500:200::b      # b.root-servers.net
master: 2001:500:2::c        # c.root-servers.net
master: 2001:500:2d::d       # d.root-servers.net
master: 2001:500:2f::f       # f.root-servers.net
master: 2001:500:12::d0d     # g.root-servers.net
master: 2001:7fd::1          # k.root-servers.net
master: 2620:0:2830:202::132 # xfr.cjr.dns.icann.org
master: 2620:0:2d0:202::132  # xfr.lax.dns.icann.org
fallback-enabled: yes
for-downstream: no
for-upstream: yes

B.3. Example Configuration: BIND 9.14

BIND 9.14 (which, at the time of publication of this document is a future release) can set up a local mirror of the root zone with a small configuration option:

zone "." {
    type mirror;
};

The simple "type mirror" configuration for the root zone works for the root zone because a default list of primary servers for the IANA root zone is built into BIND 9.14. In order to set up mirroring of any other zone, an explicit list of primary servers needs to be provided.

See the documentation for BIND 9.14 (when it is released) for more detail about how to use this simplified configuration.

B.4. Example Configuration: Unbound 1.9

Recent versions of Unbound have a "auth-zone" feature that allows local mirroring of the root zone. Configuration looks like:
auth-zone:
  name: ".
  master: "b.root-servers.net"
  master: "c.root-servers.net"
  master: "d.root-servers.net"
  master: "f.root-servers.net"
  master: "g.root-servers.net"
  master: "k.root-servers.net"
  fallback-enabled: yes
  for-downstream: no
  for-upstream: yes
  zonefile: "root.zone"

B.5. Example Configuration: Knot Resolver

Knot Resolver uses its "prefill" module to load the root zone information. This is described at <https://knot-resolver.readthedocs.io/en/stable/modules.html#root-on-loopback-rfc-7706>.


Windows Server 2012 contains a DNS server in the "DNS Manager" component. When activated, that component acts as a recursive server. DNS Manager can also act as an authoritative server.

Using this configuration, queries for information in the root zone are returned with the AA bit set.

The steps to configure DNS Manager to implement the requirements in this document are:

1. Launch the DNS Manager GUI. This can be done from the command line ("dnsmgmt.msc") or from the Service Manager (the "DNS" command in the "Tools" menu).

2. In the hierarchy under the server on which the service is running, right-click on the "Forward Lookup Zones", and select "New Zone". This brings up a succession of dialog boxes.

3. In the "Zone Type" dialog box, select "Secondary zone".

4. In the "Zone Name" dialog box, enter ".".

5. In the "Master DNS Servers" dialog box, enter "b.root-servers.net". The system validates that it can do a zone transfer from that server. (After this configuration is
completed, the DNS Manager will attempt to transfer from all of
the root zone servers.)

6. In the "Completing the New Zone Wizard" dialog box, click
"Finish".

7. Verify that the DNS Manager is acting as a recursive resolver.
Right-click on the server name in the hierarchy, choosing the
"Advanced" tab in the dialog box. See that "Disable recursion
(also disables forwarders)" is not selected, and that "Enable
DNSSEC validation for remote responses" is selected.

Acknowledgements

The authors fully acknowledge that running a copy of the root zone on
the loopback address is not a new concept, and that we have chatted
with many people about that idea over time. For example, Bill
Manning described a similar solution to the problems in his doctoral
dissertation in 2013 [Manning2013].

Evan Hunt contributed greatly to the logic in the requirements.
Other significant contributors include Wouter Wijngaards, Tony Hain,
Doug Barton, Greg Lindsay, and Akira Kato. The authors also received
many offline comments about making the document clear that this is
just a description of a way to operate a root zone on the same host,
and not a recommendation to do so.

People who contributed to this update to RFC 7706 include: Florian
Obser, nusenu, Wouter Wijngaards, [[ others go here ]].

Authors’ Addresses

Warren Kumari
Google

Email: Warren@kumari.net

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org
Address-specific DNS aliases (ANAME)
draft-ietf-dnsop-aname-03

Abstract

This document defines the "ANAME" DNS RR type, to provide similar functionality to CNAME, but only for type A and AAAA 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 October 17, 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
   1.1. Overview
   1.2. Terminology
2. The ANAME resource record
   2.1. Presentation and wire format
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 A and/or AAAA records 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.

- 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.

- The zone maintenance mechanism described in Section 5 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 Additional section processing rules in Section 3 inform resolvers that an ANAME record is in play.

- Resolvers can use this ANAME information as described in Section 6 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 4).

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 5) 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.

ANAME records 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.

Like other types, ANAME records 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. Additional section processing

The requirements in this section apply to both recursive and authoritative servers.

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.

3.1. Address queries

When a server receives an address query for a name that has an ANAME record, the response’s Additional section:

- MUST contain the ANAME record;
- MAY contain the target address records that match the query type (or the corresponding proof of nonexistence), if they are available and the target address RDATA fields differ from the sibling address RRset.

The ANAME record indicates to a client that it might wish to resolve the target address records itself. The target address records might not be available if the server is authoritative and does not include out-of-zone or non-authoritative data in its answers, or if the server is recursive and the records are not in the cache.

3.2. ANAME queries

When a server receives an query for type ANAME, there are three possibilities:
The query resolved to an ANAME record, and the server has the target address records; any target address records SHOULD be added to the Additional section.

The query resolved to an ANAME record, and the server does not have the target address records; any sibling address records SHOULD be added to the Additional section.

The query did not resolve to an ANAME record; any address records with the same owner name SHOULD be added to the Additional section of the NOERROR response.

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.

4. Substituting ANAME sibling address records

This process is used by both primary masters (see Section 5) and resolvers (see Section 6), though they vary in how they apply the edit described in the final step.

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. Replace the owner of the target address records with the owner of the ANAME record. Reduce the TTL to match the ANAME record if it is greater. 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 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.

5. 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:

- Perform ANAME sibling address record substitution as described in Section 4. 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.

- If resolution failed, wait for a period before trying again. This retry time SHOULD be configurable.

- Otherwise, wait until the target address record TTL has expired, 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 E describes how ANAME would fit in different DNS architectures that use online signing or tailored responses.

5.1. 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 E.

Secondary servers that rely on zone transfers to obtain sibling address records, just like the rest of the zone, and serve them in the usual way (with Section 3 Additional section processing if they support it). A working DNS NOTIFY [RFC1996] setup is recommended to avoid extra delays propagating updated sibling address records when they change.

5.2. DNSSEC

A zone containing ANAME records that will update A and AAAA records has to do so before signing the zone with DNSSEC [RFC4033] [RFC4034] [RFC4035].

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

5.3. 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 (or the ANAME TTL if that is smaller); however the exact mechanism for obtaining the target is unspecified, so cache effects or deliberate policies might make the sibling TTL smaller. There is a more extended discussion of TTL handling in #ttls.

6. 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 additional section, as described in Section 3. 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. It SHOULD include the target address records in the Additional section of its responses as described in Section 3.

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

- 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.)

- 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 4. Any edit performed in the final step is applied to the Answer section of the response. The resolver SHOULD then perform Additional section processing as described in Section 3.

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 4. 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.)

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 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 -03

- Grammar improvements (Olli Vanhoja)
- Split up Implications section, clarify text on zone transfers and dynamic updates.
- Rewrite Alternative setup section and move to Appendix, add text on zone transfer scalability concerns and GeoIP.

10.2. 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


11.2. Informative References


11.3. URIs

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

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"
```
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) should be set as the target address record TTL plus the sibling address record TTL.

[MM: Discuss: I think it should be just the ANAME record TTL perhaps the minimum of ANAME and sibling address RRset TTL. We should provide some guidance on TTL settings for ANAME].

[TF: see issue #30]

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:

- by directly querying an authoritative server;
- 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 re-checks 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).

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. This stops the end-to-end TTL from being inflated by each ANAME in the chain.

C.4. 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. Answer vs Additional sections

[MM: Discuss what should be in the additional section: ANAME makes sense, but differs from CNAME logic (where the CNAME is in the answer section). Additional target records that match the query type in my opinion should go in the answer section. Additional target address records that do not match the query type can go in the additional section].

[TF: from experience with DNAME I think there’s a risk of interop problems if we put unexpected records in the answer section, so I
said everything should go in additional. We’ll expand this appendix to explain the rationale.]

Appendix E. Alternative setups

If you are a large scale DNS provider, ANAME may introduce some scalability concerns. 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 4) 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 standard zone transfers, and already implement their ANAME-like processing inside the name server daemon, substituting ANAME sibling address records on demand.

Also, 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 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.

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.

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
Abstract

This document encourages the practice of permitting DNS messages to be carried over TCP on the Internet. It also considers the consequences with this form of DNS communication and the potential operational issues that can arise when this best common practice is not upheld.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 26, 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 ................................................. 3
   1.1. Requirements Language ................................ 4
2. Background .................................................. 4
   2.1. Uneven Transport Usage and Preference .................. 4
   2.2. Waiting for Large Messages and Reliability .............. 5
   2.3. EDNS0 .................................................. 5
   2.4. Fragmentation and Truncation ............................ 6
   2.5. "Only Zone Transfers Use TCP" .......................... 7
3. DNS over TCP Requirements .................................... 7
4. Network and System Considerations ............................ 8
   4.1. Connection Admission ................................... 8
   4.2. Connection Management ................................... 9
   4.3. Connection Termination .................................. 10
5. DNS over TCP Filtering Risks .................................. 10
   5.1. DNS Wedgie ............................................. 11
   5.2. DNS Root Zone KSK Rollover ............................. 11
   5.3. DNS-over-TLS ........................................... 11
6. Logging and Monitoring ........................................ 12
7. Acknowledgments .............................................. 12
8. IANA Considerations .......................................... 13
9. Security Considerations ....................................... 13
10. Privacy Considerations ....................................... 13
11. Examples .................................................... 13
12. References .................................................. 13
   12.1. Normative References .................................... 13
   12.2. Informative References .................................. 13
Appendix A. Standards Related to DNS Transport over TCP ....... 19
   A.1. IETF RFC 1035 - DOMAIN NAMES - IMPLEMENTATION AND
        SPECIFICATION ........................................... 19
   A.2. IETF RFC 1536 - Common DNS Implementation Errors and
        Suggested Fixes .......................................... 19
   A.3. IETF RFC 1995 - Incremental Zone Transfer in DNS ........ 19
   A.4. IETF RFC 1996 - A Mechanism for Prompt Notification of
        Zone Changes (DNS NOTIFY) .............................. 20
   A.5. IETF RFC 2181 - Clarifications to the DNS Specification .. 20
   A.6. IETF RFC 2694 - DNS extensions to Network Address
        Translators (DNS_ALG) .................................... 20
   A.7. IETF RFC 3225 - Indicating Resolver Support of DNSSEC ... 20
   A.8. IETF RFC 3326 - DNSSEC and IPv6 A6 aware server/resolver
        message size requirements .............................. 20
   A.9. IETF RFC 4472 - Operational Considerations and Issues
        with IPv6 DNS ........................................... 21
   A.10. IETF RFC 5452 - Measures for Making DNS More Resilient
1. Introduction

DNS messages may be delivered using UDP or TCP communications. While most DNS transactions are carried over UDP, some operators have been led to believe that any DNS over TCP traffic is unwanted or unnecessary for general DNS operation. When DNS over TCP has been restricted, a variety of communication failures and debugging challenges often arise. As DNS and new naming system features have evolved, TCP as a transport has become increasingly important for the correct and safe operation of an Internet DNS. Reflecting modern usage, the DNS standards were recently updated to declare support for TCP is now a required part of the DNS implementation.
This document is the formal requirements equivalent for the operational community, encouraging system administrators, network engineers, and security staff to ensure DNS over TCP communications support is on par with DNS over UDP communications.

1.1. Requirements Language

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 RFC 2119 [RFC2119].

2. Background

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

2.1. Uneven Transport Usage and Preference

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

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

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

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

and further stipulated:

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

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

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

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

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

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

2.3. EDNS0

In 1999 the IETF published the Extension Mechanisms for DNS (EDNS0) in [RFC2671] (superseded in 2013 by an update in [RFC6891]). This document standardized a way for communicating DNS nodes to perform rudimentary capabilities negotiation. One such capability written into the base specification and present in every ENDNS0 compatible message is the value of the maximum UDP payload size the sender can support. This unsigned 16-bit field specifies in bytes the maximum (possibly fragmented) DNS message size a node is capable of receiving. In practice, typical values are a subset of the 512 to 4096 byte range. EDNS0 became widely deployed over the next several years and numerous surveys have shown many systems currently support larger UDP MTUs [CASTRO2010], [NETALYZR] with EDNS0.

The natural effect of EDNS0 deployment meant DNS messages larger than 512 bytes would be less reliant on TCP than they might otherwise have been. While a non-negligible population of DNS systems lacked EDNS0 or fall back to TCP when necessary, DNS over TCP transactions remained a very small fraction of overall DNS traffic [VERISIGN].
2.4. Fragmentation and Truncation

Although EDNS0 provides a way for endpoints to signal support for DNS messages exceeding 512 bytes, the realities of a diverse and inconsistently deployed Internet may result in some large messages being unable to reach their destination. Any IP datagram whose size exceeds the MTU of a link it transits will be fragmented and then reassembled by the receiving host. Unfortunately, it is not uncommon for middleboxes and firewalls to block IP fragments. If one or more fragments do not arrive, the application does not receive the message and the request times out.

For IPv4-connected hosts, the de-facto MTU is often the Ethernet payload size of 1500 bytes. This means that the largest unfragmented UDP DNS message that can be sent over IPv4 is likely 1472 bytes. For IPv6, the situation is a little more complicated. First, IPv6 headers are 40 bytes (versus 20 without options in IPv4). Second, it seems as though some people have mis-interpreted IPv6’s required minimum MTU of 1280 as a required maximum. Third, fragmentation in IPv6 can only be done by the host originating the datagram. The need to fragment is conveyed in an ICMPv6 "packet too big" message. The originating host indicates a fragmented datagram with IPv6 extension headers. Unfortunately, it is quite common for both ICMPv6 and IPv6 extension headers to be blocked by middleboxes. According to [HUSTON] some 35% of IPv6-capable recursive resolvers were unable to receive a fragmented IPv6 packet.

The practical consequence of all this is that DNS requestors must be prepared to retry queries with different EDNS0 maximum message size values. Administrators of BIND are likely to be familiar with seeing "success resolving ... after reducing the advertised EDNS0 UDP packet size to 512 octets" messages in their system logs.

Often, reducing the EDNS0 UDP packet size leads to a successful response. That is, the necessary data fits within the smaller message size. However, when the data does not fit, the server sets the truncated flag in its response, indicating the client should retry over TCP to receive the whole response. This is undesirable from the client’s point of view because it adds more latency, and potentially undesirable from the server’s point of view due to the increased resource requirements of TCP.

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

2.5. "Only Zone Transfers Use TCP"

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

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

3. DNS over TCP Requirements

An average increase in DNS message size, the continued development of new DNS features [Appendix A], and a denial of service mitigation technique [Section 9] have suggested that DNS over TCP transactions are as important to the correct and safe operation of the Internet DNS as ever, if not more so. Furthermore, there has been serious research that argues connection-oriented DNS transactions may provide security and privacy advantages over UDP transport. [TDNS] In fact [RFC7858], a Standards Track document, is just this sort of specification. Therefore, this document makes explicit that it is undesirable for network operators to artificially inhibit DNS over TCP transport.

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

- Authoritative servers MUST support and service all TCP queries so that they do not limit the size of responses to what fits in a single UDP packet.
Recursive servers (or forwarders) MUST support and service all TCP queries so that they do not prevent large responses from a TCP-capable server from reaching its TCP-capable clients.

Regarding the choice of limiting the resources a server devotes to queries, Section 6.1.3.2 in [RFC1123] also says:

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

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

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

4. Network and System Considerations

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

4.1. Connection Admission

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

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

The FreeBSD operating system has an "accept filter" feature that postpones delivery of TCP connections to applications until a complete, valid request has been received. The dns_accf(9) filter ensures that a valid DNS message is received. If not, the bogus connection never reaches the application. Applications must be coded and configured to make use of this filter.

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

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

DNS clients SHOULD also enable TFO when possible. Currently, on some operating systems it is not implemented or disabled by default. [WIKIPEDIA_TFO] describes applications and operating systems that support TFO.

4.2. Connection Management

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

DNS server software SHOULD provide a configurable limit on the total number of established TCP connections. If the limit is reached, the application is expected to either close existing (idle) connections or refuse new connections. Operators SHOULD ensure the limit is configured appropriately for their particular situation.

DNS server software MAY provide a configurable limit on the number of established connections per source IP address or subnet. This can be used to ensure that a single or small set of users can not consume all TCP resources and deny service to other users. Operators SHOULD
ensure this limit is configured appropriately, based on their number of diversity of users.

DNS server software SHOULD provide a configurable timeout for idle TCP connections. For very busy name servers this might be set to a low value, such as a few seconds. For less busy servers it might be set to a higher value, such as tens of seconds. DNS clients and servers SHOULD signal their timeout values using the edns-tcp-keepalive option.[RFC7828]

DNS server software MAY provide a configurable limit on the number of transactions per TCP connection. This document does not offer advice on particular values for such a limit.

Similarly, DNS server software MAY provide a configurable limit on the total duration of a TCP connection. This document does not offer advice on particular values for such a limit.

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

4.3. Connection Termination

In general, it is preferable for clients to initiate the close of a TCP connection. The TCP peer that initiates a connection close retains the socket in the TIME_WAIT state for some amount of time, possibly a few minutes. On a busy server, the accumulation of many sockets in TIME_WAIT can cause performance problems or even denial of service.

On systems where large numbers of sockets in TIME_WAIT are observed, it may be beneficial to tune the local TCP parameters. For example, the Linux kernel provides a number of "sysctl" parameters related to TIME_WAIT, such as net.ipv4.tcp_fin_timeout, net.ipv4.tcp_tw_recycle, and net.ipv4.tcp_tw_reuse. In extreme cases, implementors and operators of very busy servers may find it necessary to utilize the SO_LINGER socket option ([Stevens] Section 7.5) with a value of zero so that the server doesn’t accumulate TIME_WAIT sockets.

5. DNS over TCP Filtering Risks

Networks that filter DNS over TCP risk losing access to significant or important pieces of the DNS name space. For a variety of reasons a DNS answer may require a DNS over TCP query. This may include large message sizes, lack of EDNS0 support, DDoS mitigation techniques, or perhaps some future capability that is as yet unforeseen will also demand TCP transport.
For example, [RFC7901] describes a latency-avoiding technique that sends extra data in DNS responses. This makes responses larger and potentially increases the risk of DDoS reflection attacks. The specification mandates the use of TCP or DNS Cookies.[RFC7873]

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

5.1. DNS Wedgie

Networks that filter DNS over TCP may inadvertently cause problems for third party resolvers as experienced by [TOYAMA]. If for instance a resolver receives a truncated answer from a server, but when the resolver resends the query using TCP and the TCP response never arrives, not only will a complete answer be unavailable, but the resolver will incur the full extent of TCP retransmissions and time outs. This situation might place extreme strain on resolver resources. If the number and frequency of these truncated answers are sufficiently high, the steady-state of lost resources as a result is a "DNS" wedgie". A DNS wedgie is generally not easily or completely mitigated by the affected DNS resolver operator.

5.2. DNS Root Zone KSK Rollover

Recent plans for a new root zone DNSSEC KSK have highlighted a potential problem in retrieving the keys.[LEWIS] Some packets in the KSK rollover process will be larger than 1280 bytes, the IPv6 minimum MTU for links carrying IPv6 traffic.[RFC2460] While studies have shown that problems due to fragment filtering or an inability to generate and receive these larger messages are negligible, any DNS server that is unable to receive large DNS over UDP messages or perform DNS over TCP may experience severe disruption of DNS service if performing DNSSEC validation.

TODO: Is this "overcome by events" now? We’ve had 1414 byte DNSKEY responses at the three ZSK rollover periods since KSK-2017 became published in the root zone.

5.3. DNS-over-TLS

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

The use of TLS places even strong operational burdens on DNS clients and servers. Cryptographic functions for authentication and encryption require additional processing. Unoptimized connection setup takes two additional round-trips compared to TCP, but can be reduced with Fast TLS connection resumption [RFC5077] and TLS False Start [RFC7918].

6. Logging and Monitoring

Developers of applications that log or monitor DNS are advised to not ignore TCP because it is rarely used or because it is hard to process. Operators are advised to ensure that their monitoring and logging applications properly capture DNS-over-TCP messages. Otherwise, attacks, exfiltration attempts, and normal traffic may go undetected.

DNS messages over TCP are in no way guaranteed to arrive in single segments. In fact, a clever attacker may attempt to hide certain messages by forcing them over very small TCP segments. Applications that capture network packets (e.g., with libpcap) should be prepared to implement and perform full TCP segment reassembly. dnscap [dnscap] is an open-source example of a DNS logging program that implements TCP reassembly.

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

7. Acknowledgments

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

The following individuals provided an array of feedback to help improve this document: Sara Dickinson, Bob Harold, Tatuya Jinmei, and Paul Hoffman. The authors are also indebted to the contributions stemming from discussion in the tcpm working group meeting at IETF 104. Any remaining errors or imperfections are the sole responsibility of the document authors.
8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

Ironically, returning truncated DNS over UDP answers in order to induce a client query to switch to DNS over TCP has become a common response to source address spoofed, DNS denial-of-service attacks [RRL]. Historically, operators have been wary of TCP-based attacks, but in recent years, UDP-based flooding attacks have proven to be the most common protocol attack on the DNS. Nevertheless, a high rate of short-lived DNS transactions over TCP may pose challenges. While many operators have provided DNS over TCP service for many years without duress, past experience is no guarantee of future success.

DNS over TCP is not unlike many other Internet TCP services. TCP threats and many mitigation strategies have been well documented in a series of documents such as [RFC4953], [RFC4987], [RFC5927], and [RFC5961].

10. Privacy Considerations

TODO: Does this document warrant privacy considerations?

11. Examples

Suggestion from IETF104 to include example config snippets ala 7706.

12. References

12.1. Normative References


12.2. Informative References


[RRL] Vixie, P. and V. Schryver, "DNS Response Rate Limiting (DNS RRL)", ISC-TN 2012-1 Draft1, April 2012.
Appendix A. Standards Related to DNS Transport over TCP

This section enumerates all known IETF RFC documents that are currently of status standard, informational, best common practice or experimental and either implicitly or explicitly make assumptions or statements about the use of TCP as a transport for the DNS germane to this document.

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

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

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

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

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

The [RFC1995] standards track document documents the use of TCP as the fallback transport when IXFR responses do not fit into a single UDP response. As with AXFR, IXFR messages are typically delivered over TCP by default in practice. XXX: is this an accurate statement?
A.4. IETF RFC 1996 - A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)

The [RFC1996] standards track document suggests a zone master may decide to issue NOTIFY messages over TCP. In practice NOTIFY messages are generally sent over UDP, but this specification leaves open the possibility that the choice of transport protocol is up to the master, and therefore a slave ought to be able to operate over both UDP and TCP.

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

The [RFC2181] standards track document includes clarifying text on how a client should react to the TC flag set on responses. It is advised the the response should be discarded and the query resent using TCP.

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

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

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

The [RFC3225] standards track document makes statements indicating DNS over TCP is "detrimental" as a result of increased traffic, latency, and server load. This document is a companion to the next document in the RFC series expressing the requirement for EDNS0 support for DNSSEC.

A.8. IETF RFC 3326 - DNSSEC and IPv6 A6 aware server/resolver message size requirements

The [RFC3226] standards track document, although updated by later DNSSEC strongly argued in favor of UDP messages over TCP largely for performance reasons. The document declares EDNS0 a requirement for DNSSEC servers and advocated packet fragmentation may be preferable to TCP in certain situations.
A.9. IETF RFC 4472 - Operational Considerations and Issues with IPv6 DNS

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

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

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

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

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

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

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

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

The [RFC5936] standards track document provides a detailed specification for the zone transfer protocol, as originally outlined in the early DNS standards. AXFR operation is limited to TCP and not specified for UDP. This document discusses TCP usage at length.
A.14. IETF RFC 5966 - DNS Transport over TCP - Implementation Requirements

This standards track document [RFC5966] instructs DNS implementers to provide support for carrying DNS over TCP messages in their software. The authors explicitly make no recommendations to operators, which we seek to address here.

A.15. IETF RFC 6304 - AS112 Nameserver Operations

[RFC6304] is an informational document enumerating the requirements for operation of AS112 project DNS servers. New AS112 nodes are tested for their ability to provide service on both UDP and TCP transports, with the implication that TCP service is an expected part of normal operations.

A.16. IETF RFC 6762 - Multicast DNS

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

A.17. IETF RFC 6891 - Extension Mechanisms for DNS (EDNS(0))

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

A.18. IETF RFC 6950 - Architectural Considerations on Application Features in the DNS

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

A.19. IETF RFC 7477 - Child-to-Parent Synchronization in DNS

This standards track document [RFC7477] specifies a RRType and protocol to signal and synchronize NS, A, and AAAA resource record changes from a child to parent zone. Since this protocol may require multiple requests and responses, it recommends utilizing DNS over TCP to ensure the conversation takes place between a consistent pair of end nodes.
A.20. IETF RFC 7720 - DNS Root Name Service Protocol and Deployment Requirements

This best current practice [RFC7720] declares root name service "MUST support UDP [RFC768] and TCP [RFC793] transport of DNS queries and responses."

A.21. IETF RFC 7766 - DNS Transport over TCP - Implementation Requirements

The standards track document [RFC7766] might be considered the direct ancestor of this operational requirements document. The implementation requirements document codifies mandatory support for DNS over TCP in compliant DNS software.

A.22. IETF RFC 7828 - The edns-tcp-keepalive EDNS0 Option

This standards track document [RFC7828] defines an EDNS0 option to negotiate an idle timeout value for long-lived DNS over TCP connections. Consequently, this document is only applicable and relevant to DNS over TCP sessions and between implementations that support this option.

A.23. IETF RFC 7858 - Specification for DNS over Transport Layer Security (TLS)

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

A.24. IETF RFC 7873 - Domain Name System (DNS) Cookies

This standards track document [RFC7873] describes an EDNS0 option to provide additional protection against query and answer forgery. This specification mentions DNS over TCP as a reasonable fallback mechanism when DNS Cookies are not available. The specification does make mention of DNS over TCP processing in two specific situations. In one, when a server receives only a client cookie in a request, the server should consider whether the request arrived over TCP and if so, it should consider accepting TCP as sufficient to authenticate the request and respond accordingly. In another, when a client receives a BADCOOKIE reply using a fresh server cookie, the client should retry using TCP as the transport.
A.25. IETF RFC 7901 - CHAIN Query Requests in DNS

This experimental specification [RFC7901] describes an EDNS0 option that can be used by a security-aware validating resolver to request and obtain a complete DNSSEC validation path for any single query. This document requires the use of DNS over TCP or a source IP address verified transport mechanism such as EDNS-COOKIE.[RFC7873]

A.26. IETF RFC 8027 - DNSSEC Roadblock Avoidance

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

A.27. IETF RFC 8094 - DNS over Datagram Transport Layer Security (DTLS)

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

A.28. IETF RFC 8162 - Using Secure DNS to Associate Certificates with Domain Names for S/MIME

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

A.29. IETF RFC 8324 - DNS Privacy, Authorization, Special Uses, Encoding, Characters, Matching, and Root Structure: Time for Another Look?

An informational document [RFC8324] that briefly discusses the common role and challenges of DNS over TCP throughout the history of DNS.
An experimental document [RFC8467] reminds implementers to consider the underlying transport protocol (e.g. TCP) when calculating the padding length when artificially increasing the DNS message size with an EDNS(0) padding option.

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

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

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

Authors’ Addresses

John Kristoff
DePaul University
Chicago, IL 60604
US

Phone: +1 312 493 0305
Email: jtk@depaul.edu
URI: https://aharp.iorc.depaul.edu
Abstract

Many enterprises today employ the service of multiple DNS providers to distribute their authoritative DNS service. Deploying DNSSEC in such an environment may present some challenges depending on the configuration and feature set in use. This document will present several deployment models that may be suitable.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of
Many enterprises today employ the service of multiple DNS providers to distribute their authoritative DNS service. This allows the DNS service to survive a complete failure of any single provider. Additionally, enterprises or providers occasionally have requirements that preclude standard zone transfer techniques [RFC1995] [RFC5936]: either non-standardized DNS features are in use that are incompatible with zone transfer, or operationally a provider must be able to (re)sign DNS records using their own keys. This document outlines some possible models of DNSSEC [RFC4033] [RFC4034] [RFC4035] deployment in such an environment.
2. Deployment Models

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

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

2.1. Multiple Signer models

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

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

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

The following descriptions consider the case of two DNS providers,
but the model is generalizable to any number.

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

- Zone owner holds the KSK, manages the DS record, and is
  responsible for signing the DNSKEY RRset and distributing the
  signed DNSKEY RRset to the providers.

- Each provider has their own ZSK which is used to sign data.

- Providers have an API that owner uses to query the ZSK public key,
  and insert a combined DNSKEY RRset that includes both ZSKs and the
  KSK, signed by the KSK.

- Note that even if the contents of the DNSKEY RRset don’t change,
  the Zone owner of course needs to periodically re-sign it as
  signature expiration approaches. The provider API is also used to
  thus periodically redistribute the refreshed DNSKEY RRset.

- Key rollovers need coordinated participation of the zone owner to
  update the DNSKEY RRset (for KSK or ZSK), and the DS RRset (for
  KSK).

2.1.2. Model 2: Unique KSK and ZSK per provider

- Each provider has their own KSK and ZSK.

- Each provider offers an API that the Zone Owner uses to import the
  ZSK of the other provider into their DNSKEY RRset.

- DNSKEY RRset is signed independently by each provider using their
  own KSK.

- Zone Owner manages the DS RRset that includes both KSKs.

- Key rollovers need coordinated participation of the zone owner to
  update the DS RRset (for KSK), and the DNSKEY RRset (for ZSK).

3. Validating Resolver Behavior

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

- The validating resolver follows a referral (delegation) to the zone in question.
- It retrieves the zone’s DNSKEY RRset from one of provider A’s nameservers.
- At some point in time, the resolver attempts to resolve a name in the zone, while the DNSKEY RRset received from provider A is still viable in its cache.
- It queries one of provider B’s nameservers to resolve the name, and obtains a response that is signed by provider B’s ZSK, which it cannot authenticate because this ZSK is not present in its cached DNSKEY RRset for the zone that it received from provider A.
- The resolver will not accept this response. It may still be able to ultimately authenticate the name by querying other nameservers for the zone until it elicits a response from one of provider A’s nameservers. But it has incurred the penalty of additional roundtrips with other nameservers, with the corresponding latency and processing costs. The exact number of additional roundtrips depends on details of the resolver’s nameserver selection algorithm and the number of nameservers configured at provider B.
- It may also be the case that a resolver is unable to provide an authenticated response because it gave up after a certain number of retries or a certain amount of delay. Or that downstream clients of the resolver that originated the query timed out waiting for a response.

Zone owners will want to deploy a DNS service that responds as efficiently as possible with validatable answers only, and hence it is important that the DNSKEY RRset at each provider is maintained with the active ZSKs of all participating providers. This ensures that resolvers can validate a response no matter which provider’s nameservers it came from.

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

4. Signing Algorithm Considerations

It is RECOMMENDED that the providers use a common signing algorithm (and common key sizes for algorithms that support variable key sizes). This ensures that the multiple providers have identical security postures and no provider is more vulnerable to cryptanalytic attack than the others.

It may however be possible to deploy a configuration where different providers use different signing algorithms. The main impediment is that current DNSSEC specifications require that if there are multiple algorithms in the DNSKEY RRset, then RRsets in the zone need to be signed with at least one DNSKEY of each algorithm, as described in RFC 4035 [RFC4035], Section 2.2. However RFC 6781 [RFC6781], Section 4.1.4, also describes both a conservative and liberal interpretation of this requirement. When validating DNS resolvers follow the liberal approach, they do not expect that zone RRsets are signed by every signing algorithm in the DNSKEY RRset, and responses with single algorithm signatures can be validated correctly assuming a valid chain of trust exists. In fact, testing by the .BR Top Level domain for their recent algorithm rollover [BR-ROLLOVER], demonstrates that the liberal approach does in fact work with current resolvers deployed on the Internet.

5. Authenticated Denial Considerations

Authenticated denial of existence enables a resolver to validate that a record does not exist. For this purpose, an authoritative server presents, in a response to the resolver, NSEC (Section 3.1.3 of [RFC4035]) or NSEC3 (Section 7.2 of [RFC5155]) records. The NSEC3 method enhances NSEC by providing opt-out for signing insecure delegations and also adds limited protection against zone enumeration attacks.

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

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

5.1. Single Method

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

5.2. Mixing Methods

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

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

In case all providers cannot be configured for a matching authenticated denial, it is advised to find lowest number of possible configurations possible across all used providers.

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

6. Key Rollover Considerations

The Multiple Signer (Section 2.1) models introduce some new requirements for DNSSEC key rollovers. Since this process necessarily involves coordinated actions on the part of providers and the Zone Owner, one reasonable strategy is for the Zone Owner to initiate key rollover operations. But other operationally plausible models may also suit, such as a DNS provider initiating a key rollover and signaling their intent to the Zone Owner in some manner.
6.1. Model 1: Common KSK, Unique ZSK per provider

- **Key Signing Key Rollover:** In this model, the two managed DNS providers share a common KSK which is held by the Zone Owner. To initiate the rollover, the Zone Owner generates a new KSK and obtains the DNSKEY RRset of each DNS provider using their respective APIs. The new KSK is added to each provider’s DNSKEY RRset and the RRset is re-signed with both the new and the old KSK. The Zone Owner then updates the DS RRset in the parent zone to point to the new KSK, and after the necessary DS record TTL period has expired, proceeds with updating the DNSKEY RRSet to remove the old KSK.

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

6.2. Model 2: Unique KSK and ZSK per provider

- **Key Signing Key Rollover:** In Model 2, each managed DNS provider has their own KSK. A KSK roll for provider A does not require any change in the DNSKEY RRset of provider B, but does require co-ordination with the Zone Owner in order to get the DS record set in the parent zone updated. The KSK roll starts with Provider A generating a new KSK and including it in their DNSKEY RRSet. The DNSKey RRset would then be signed by both the new and old KSK.
The new KSK is communicated to the Zone Owner, after which the Zone Owner updates the DS RRset to replace the DS record for the old KSK with a DS record for the new KSK. After the necessary DS RRset TTL period has elapsed, the old KSK can be removed from provider A’s DNSKEY RRset.

- Zone Signing Key Rollover: In Model 2, each managed DNS provider has their own ZSK. The ZSK roll for provider A would start with them generating new ZSK and including it in their DNSKEY RRset and re-signing the new DNSKEY RRset with their KSK. The new ZSK of provider A would then be communicated to the Zone Owner, who will initiate the process of importing this ZSK into the DNSKEY RRsets of the other providers, using their respective APIs. Once the necessary Pre-Publish key rollover time periods have elapsed, provider A and the Zone Owner can initiate the process of removing the old ZSK from the DNSKEY RRset of all providers.

7. Inter Provider Handoff

The primary use case for the models presented in this draft are for steady state operation of multiple concurrent signing providers. But they can also be leveraged in a fairly straightforward manner to perform non-disruptive transfer of a signed DNS domain from one provider to another. This involves initially bringing the new provider into a multi-provider configuration, and then at a later time detaching the old provider. [TBD: flesh out this use case in more detail.]

8. Key Management Mechanism Requirements

Managed DNS providers often have their own proprietary zone configuration and data management APIs, typically utilizing HTTPS/REST interfaces. So, rather than outlining a new API for key management here, we describe the specific functions that the provider API needs to support in order to enable the multi-signer models. The Zone owner is expected to use these API functions to perform key management tasks. Other mechanisms that can offer these functions, if supported by the providers, include the DNS UPDATE protocol [RFC2136] and EPP [RFC5731].

- The API must offer a way to query the current DNSKEY RRset of the provider
- For model 1, the API must offer a way to import a signed DNSKEY RRset and replace the current one at the provider.
- For model 2, the API must offer a way to import a DNSKEY record from an external provider into the current DNSKEY RRset
In model 2, once initially bootstrapped with each others zone signing keys via these API mechanisms, providers could, if desired, periodically query each others DNSKEY RRs and automatically import or withdraw ZSKs in the keyset as key rollover events happen.

9. IANA Considerations

This document includes no request to IANA.

10. Security Considerations

The Zone key import APIs required by these models need to be strongly authenticated to prevent tampering of key material by malicious third parties. Many providers today offer REST/HTTPS APIs that utilize a number of authentication mechanisms (username/password, API keys etc). If DNS protocol mechanisms like UPDATE are being used for key insertion and deletion, they should similarly be strongly authenticated, e.g. by employing Transaction Signatures (TSIG) [RFC2845].

11. Acknowledgments

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

12. References

12.1. Normative References


Internet-Draft        Multi Provider DNSSEC models            March 2019


12.2. Informative References


Authors' Addresses

Shumon Huque
Salesforce

Email: shuque@gmail.com

Pallavi Aras
Salesforce

Email: paras@salesforce.com

John Dickinson
Sinodun

Email: jad@sinodun.com

Jan Vcelak
NS1

Email: jvcelak@ns1.com

David Blacka
Verisign

Email: davidb@verisign.com
YANG Types for DNS Classes and Resource Record Types
draft-lhotka-dnsop-iana-class-type-yang-01

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the
provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering
Task Force (IETF). Note that other groups may also distribute
working documents as Internet-Drafts. The list of current Internet-
Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months
and may be updated, replaced, or obsoleted by other documents at any
time. It is inappropriate to use Internet-Drafts as reference
material or to cite them other than as "work in progress."

This Internet-Draft will expire on 31 August 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 (http://trustee.ietf.org/
license-info) in effect on the date of publication of this document.
Please review these documents carefully, as they describe your rights
and restrictions with respect to this document. Code Components
extracted from this document must include Simplified BSD License text
as described in Section 4.e of the Trust Legal Provisions and are
provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction
2. YANG Design Considerations
3. YANG Module
4. IANA Considerations
   4.1. URI Registrations
   4.2. YANG Module Registrations
5. Security Considerations
6. References
   6.1. Normative References
   6.2. Informative References
Authors’ Addresses

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

* authoritative servers
* resolvers
* zone data.

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

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

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

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

2. YANG Design Considerations

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

[RFC3597] introduced the option of specifying a class or type via its assigned decimal number, as an alternative to the mnemonic name. For example, the "IN" class can be equivalently written as "CLASS1", and "AAAA" type as "TYPE28".
Accordingly, the derived types "dns-class" and "rr-type" are defined in the YANG module as a union of two member types:

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

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

3. YANG Module

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

<CODE BEGINS> file "iana-dns-class-rr-type@2019-02-27.yang"
module iana-dns-class-rr-type {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-dns-class-rr-type";
  prefix dnsct;
  organization "Internet Assigned Numbers Authority (IANA)";
  contact "Internet Assigned Numbers Authority
               Postal: ICANN
               4676 Admiralty Way, Suite 330
               Marina del Rey, CA 90292
               Tel: +1 310 823 9358
               <mailto:iana@iana.org>";
  description "This YANG module translates IANA registries ‘DNS CLASSes’ and ‘Resource Record (RR) TYPEs’ to YANG derived types.

  Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC XXXX (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for full legal notices."
  reference "IANA ‘Domain Name System (DNS) Parameters’ registry
             https://www.iana.org/assignments/dns-parameters";
  revision 2019-02-27 {
    description "Initial revision.";
  }
}</CODE ENDS>
typedef dns-class-name {
  type enumeration {
    enum IN {
      value "1";
      description "Internet";
      reference "RFC 1035: Domain Names - Implementation and Specification";
    }
    enum CH {
      value "3";
      description "Chaos";
      reference "Moon, D., 'Chaosnet', A. I. Memo 628, MIT Artificial Intelligence Laboratory, June 1981";
    }
    enum HS {
      value "4";
      description "Hesiod";
    }
    enum NONE {
      value "254";
      description "QCLASS NONE";
      reference "RFC 2136: Dynamic Updates in the Domain Name System (DNS UPDATE)";
    }
    enum ANY {
      value "255";
      description "QCLASS * (ANY)";
      reference "RFC 1035: Domain Names - Implementation and Specification";
    }
  }
}
description "This enumeration type defines mnemonic names and corresponding numeric values of DNS classes.";
reference "RFC 6895: Domain Name System (DNS) IANA Considerations";

typedef dns-class {
  type union {
    type uint16;
    type dns-class-name;
  }
}
description "This type allows for referring to a DNS class using either the assigned mnemonic name or numeric value.";
typedef rr-type-name {
    type enumeration {
        enum A {
            value "1";
            description "A host address.";
            reference "RFC 1035: Domain Names - Implementation and Specification";
        }
        enum NS {
            value "2";
            description "An authoritative name server.";
            reference "RFC 1035: Domain Names - Implementation and Specification";
        }
        enum MD {
            value "3";
            status "obsolete";
            description "A mail destination (obsolete use MX).";
            reference "RFC 1035: Domain Names - Implementation and Specification";
        }
        enum MF {
            value "4";
            status "obsolete";
            description "A mail forwarder (obsolete use MX).";
            reference "RFC 1035: Domain Names - Implementation and Specification";
        }
        enum CNAME {
            value "5";
            description "The canonical name for an alias.";
            reference "RFC 1035: Domain Names - Implementation and Specification";
        }
        enum SOA {
            value "6";
            description "Start of a zone of authority.";
            reference "RFC 1035: Domain Names - Implementation and Specification";
        }
        enum MB {
            value "7";
            description "A mailbox domain name (experimental).";
            reference "RFC 1035: Domain Names - Implementation and Specification";
        }
        enum MG {
            value "8";
            description "A mail group member (experimental).";
            reference "RFC 1035: Domain Names - Implementation and Specification";
        }
    }
}
enum MR {
    value "9";
    description
        "A mail rename domain name (experimental).";
    reference
        "RFC 1035: Domain Names - Implementation and Specification";
}
enum NULL {
    value "10";
    description
        "A null RR (experimental).";
    reference
        "RFC 1035: Domain Names - Implementation and Specification";
}
enum WKS {
    value "11";
    description
        "A well known service description.";
    reference
        "RFC 1035: Domain Names - Implementation and Specification";
}
enum PTR {
    value "12";
    description
        "A domain name pointer.";
    reference
        "RFC 1035: Domain Names - Implementation and Specification";
}
enum HINFO {
    value "13";
    description
        "Host information.";
    reference
        "RFC 1035: Domain Names - Implementation and Specification";
}
enum MINFO {
    value "14";
    description
        "Mailbox or mail list information.";
    reference
        "RFC 1035: Domain Names - Implementation and Specification";
}
enum MX {
    value "15";
    description
        "Mail exchange.";
    reference
        "RFC 1035: Domain Names - Implementation and Specification";
}
enum TXT {
    value "16";
    description
        "Text strings.";
    reference
        "RFC 1035: Domain Names - Implementation and Specification";
}
enum RP {
    value "17";
}
enum AFSDB {
  value "18";
  description "AFS data base location.";
  reference "RFC 1183: New DNS RR Definitions"
}

enum X25 {
  value "19";
  description "X.25 PSDN address.";
  reference "RFC 1183: New DNS RR Definitions"
}

enum ISDN {
  value "20";
  description "ISDN address.";
  reference "RFC 1183: New DNS RR Definitions"
}

enum RT {
  value "21";
  description "Route through.";
  reference "RFC 1183: New DNS RR Definitions"
}

enum NSAP {
  value "22";
  description "NSAP address, NSAP style A record.";
  reference "RFC 1706: DNS NSAP Resource Records"
}

enum NSAP-PTR {
  value "23";
  description "Domain name pointer, NSAP style.";
  reference "RFC 1348: DNS NSAP RRs"
    - RFC 1637: DNS NSAP Resource Records
    - RFC 1706: DNS NSAP Resource Records"
}

enum SIG {
  value "24";
  description "Security signature.";
  reference "RFC 4034: Resource Records for the DNS Security Extensions"
    - RFC 3755: Legacy Resolver Compatibility for Delegation Signer (DS)
    - RFC 2535: Domain Name System Security Extensions
    - RFC 2536: DSA KEYS and SIGs in the Domain Name System
- RFC 2537: RSA/MD5 KEYS and SIGs in the Domain Name System (DNS)

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

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

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

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

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

- RFC 2535: Domain Name System Security Extensions

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

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

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

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

- RFC 3110: RSA/SHA-1 SIGs and RSA KEYS in the Domain Name System (DNS)"
};

} enum PX {
    value "26";
    description "X.400 mail mapping information."
    reference "RFC 2163: Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM)"
};

} enum GPOS {
    value "27";
    description "Geographical position."
    reference "RFC 1712: DNS Encoding of Geographical Location"
};

} enum AAAA {
    value "28";
    description "IPv6 address."
    reference "RFC 3596: DNS Extensions to Support IP Version 6"
};

} enum LOC {
    value "29";
    description "..."
"Location information.";
reference
"RFC 1876: A Means for Expressing Location Information in 
the Domain Name System";
}
enum NXT {
  value "30";
  status "obsolete";
  description
  "Next domain (obsolete).";
  reference
  "- RFC 3755: Legacy Resolver Compatibility for Delegation 
  Signer (DS)
  - RFC 2535: Domain Name System Security Extensions";
}
enum EID {
  value "31";
  description
  "Endpoint identifier.";
}
enum NIMLOC {
  value "32";
  description
  "Nimrod locator.";
}
enum SRV {
  value "33";
  description
  "Server selection.";
  reference
  "RFC 2782: A DNS RR for specifying the location of services 
  (DNS SRV)";
}
enum ATMA {
  value "34";
  description
  "ATM address.";
  reference
  "ATM Forum Technical Committee, ’ATM Name System V2.0’, 
  AF-DANS-0152.00, July 2000";
}
enum NAPTR {
  value "35";
  description
  "Naming authority pointer.";
  reference
  "- RFC 2915: The Naming Authority Pointer (NAPTR) DNS 
  Resource Record
  - RFC 2168: Resolution of Uniform Resource Identifiers 
    using the Domain Name System
  - RFC 3403: Dynamic Delegation Discovery System (DDDS) 
    Part Three: The Domain Name System (DNS) Database";
}
enum KX {
  value "36";
  description
  "Key exchanger.";
  reference
  "RFC 2230: Key Exchange Delegation Record for the DNS";
}
enum CERT {
  value "37";
  description
  "Certificate.";
}
reference
   "RFC 4398: Storing Certificates in the Domain Name System (DNS)");

} enum A6 {
   value "38";
   status "obsolete";
   description
      "IPv6 address (obsolete use AAAA)."
   reference
      "- RFC 3226: DNSSEC and IPv6 A6 Aware Server/Resolver Message Size Requirements
      - RFC 2874: DNS Extensions to Support IPv6 Address Aggregation and Renumbering
      - RFC 6563: Moving A6 to Historic Status"
}

} enum DNAME {
   value "39";
   description
      "DNAME".
   reference
      "- RFC 2672: Non-Terminal DNS Name Redirection
      - RFC 6672: DNAME Redirection in the DNS"
}

} enum SINK {
   value "40";
   description
      "Kitchen sink."
}

} enum OPT {
   value "41";
   description
      "OPT pseudo-RR.".
   reference
      "- RFC 6891: Extension Mechanisms for DNS (EDNS(0))
      - RFC 3225: Indicating Resolver Support of DNSSEC"
}

} enum APL {
   value "42";
   description
      "Address prefix list.";
   reference
      "RFC 3123: A DNS RR Type for Lists of Address Prefixes (APL RR)"
}

} enum DS {
   value "43";
   description
      "Delegation signer.";
   reference
      "- RFC 4034: Resource Records for the DNS Security Extensions
      - RFC 3658: Delegation Signer (DS) Resource Record (RR)"
}

} enum SSHFP {
   value "44";
   description
      "SSH key fingerprint.";
   reference
      "RFC 4255: Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints";
enum IPSECKEY {
    value "45";
    description
        "IPSec key.";
    reference
        "RFC 4025: A Method for Storing IPsec Keying Material in DNS";
}

enum RRSIG {
    value "46";
    description
        "RR signature.";
    reference
        "- RFC 4034: Resource Records for the DNS Security Extensions
          - RFC 3755: Legacy Resolver Compatibility for Delegation Signer (DS)";
}

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

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

enum DHCID {
    value "49";
    description
        "DHCID resource record.";
    reference
        "RFC 4701: A DNS Resource Record (RR) for Encoding Dynamic Host Configuration Protocol (DHCP) Information (DHCID RR)";
}

enum NSEC3 {
    value "50";
    description
        "NSEC3 resource record.";
    reference
        "RFC 5155: DNS Security (DNSSEC) Hashed Authenticated Denial of Existence";
}

enum NSEC3PARAM {
    value "51";
    description
        "NSEC3PARAM resource record.";
    reference
        "RFC 5155: DNS Security (DNSSEC) Hashed Authenticated Denial of Existence";
}

enum TLSA {

value "52";
description
"TLSA resource record.";
reference
}
enum SMIMEA {
  value "53";
description
"S/MIME cert association";
reference
"RFC 8162: Using Secure DNS to Associate Certificates with Domain Names for S/MIME";
}
enum HIP {
  value "55";
description
"Host identity protocol.";
reference
"RFC 5205: Host Identity Protocol (HIP) Domain Name System (DNS) Extension";
}
enum NINFO {
  value "56";
description
"NINFO resource record.";
}
enum RKEY {
  value "57";
description
"RKEY resource record.";
}
enum TALINK {
  value "58";
description
"Trust anchor LINK.";
}
enum CDS {
  value "59";
description
"Child DS.";
reference
"RFC 7344: Automating DNSSEC Delegation Trust Maintenance";
}
enum CDNSKEY {
  value "60";
description
"DNSKEY(s) the child wants reflected in DS.";
reference
"RFC 7344: Automating DNSSEC Delegation Trust Maintenance";
}
enum OPENPGPKEY {
  value "61";
description
"OpenPGP key.";
reference
"RFC 7929: DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP";
}
enum CSYNC {
  value "62";
description
"Child-to-parent synchronization.";
reference
enum SPF {
    value "99";
    description "SPF (sender policy framework) resource record.";
    reference "RFC 7208: Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1";
}

enum UINFO {
    value "100";
    description "IANA-reserved.";
}

enum UID {
    value "101";
    description "IANA-reserved.";
}

enum GID {
    value "102";
    description "IANA-reserved.";
}

enum UNSPEC {
    value "103";
    description "IANA-reserved.";
}

enum NID {
    value "104";
    description "Node identifier.";
}

enum L32 {
    value "105";
    description "L32 resource record.";
}

enum L64 {
    value "106";
    description "L64 resource record.";
}

enum LP {
    value "107";
    description "LP resource record.";
}

enum EUI48 {
    value "108";
    description "An EUI-48 address.";
    reference
"RFC 7043: Resource Records for EUI-48 and EUI-64 Addresses in the DNS";
}
enum EUI64 {
  value "109";
  description "An EUI-64 address.";
  reference "RFC 7043: Resource Records for EUI-48 and EUI-64 Addresses in the DNS";
}
enum TKEY {
  value "249";
  description "Transaction key.";
  reference "RFC 2930: Secret Key Establishment for DNS (TKEY RR)";
}
enum TSIG {
  value "250";
  description "Transaction signature.";
  reference "RFC 2845: Secret Key Transaction Authentication for DNS (TSIG)";
}
enum IXFR {
  value "251";
  description "Incremental transfer.";
  reference "RFC 1995: Incremental Zone Transfer in DNS";
}
enum AXFR {
  value "252";
  description "Transfer of an entire zone.";
  reference "- RFC 1035: Domain Names - Implementation and Specification
- RFC 5936: DNS Zone Transfer Protocol (AXFR)";
}
enum MAILB {
  value "253";
  description "Mailbox-related RRs (MB, MG or MR).";
  reference "RFC 1035: Domain Names - Implementation and Specification";
}
enum MAILA {
  value "254";
  status "obsolete";
  description "Mail agent RRs (obsolete see MX).";
  reference "RFC 1035: Domain Names - Implementation and Specification";
}
enum * {
  value "255";
  description "A request for all records the server/cache has available.";
  reference "- RFC 1035: Domain Names - Implementation and
Specification
- RFC 6895: Domain Name System (DNS) IANA Considerations;

enum URI {
    value "256";
    description
        "URI resource record.";
    reference
        "RFC 7553: The Uniform Resource Identifier (URI) DNS Resource Record";
}

enum CAA {
    value "257";
    description
        "Certification authority authorization.";
    reference
        "RFC 6844: DNS Certification Authority Authorization (CAA) Resource Record";
}

enum AVC {
    value "258";
    description
        "Application visibility and control.";
}

enum DOA {
    value "259";
    description
        "Digital object architecture";
    reference
        "draft-durand-doa-over-dns: DOA over DNS";
}

enum TA {
    value "32768";
    description
        "DNSSEC trust authorities.";
}

enum DLV {
    value "32769";
    description
        "DNSSEC lookaside validation.";
    reference
        "RFC 4431: The DNSSEC Lookaside Validation (DLV) DNS Resource Record";
}

description
    "This enumeration type defines mnemonic names and corresponding numeric values of DNS resource record types.";
    reference
        "- RFC 6895: Domain Name System (DNS) IANA Considerations
        - RFC 1035: Domain Names - Implementation and Specification";
}

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

<CODE ENDS>
4. IANA Considerations

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

OLD:
  [RFC6895]

NEW:
  [RFC6895][RFCXXXX]

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

OLD:

4.1. URI Registrations

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

Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

4.2. YANG Module Registrations

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

name: iana-dns-class-rr-type
prefix: dnsct
reference: RFC XXXX

5. Security Considerations

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

6. References

6.1. Normative References


6.2. Informative References


Authors’ Addresses

Ladislav Lhotka
CZ.NIC
Email: lhotka@nic.cz

Petr Spacek
CZ.NIC
Email: petr.spacek@nic.cz
Responsibility for Authoritative DNS and DNSSEC Mistakes
draft-livingood-dnsop-auth-dnssec-mistakes-04

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 22, 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
1. Introduction

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

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

Operators of DNS recursive resolvers, including Internet Service Providers (ISPs) and cloud-based DNS resolvers, occasionally observe
domains incorrectly managing DNSSEC-related resource records. This mismanagement triggers DNSSEC validation failures, and then causes large numbers of end users to be unable to reach a domain. Similarly, errors in non-DNSSEC-related authoritative DNS resource records result in failures, from NXDOMAIN responses to valid responses containing outdated or unreachable hosts.

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

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

2. Domain Validation Failures

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

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

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

3. Responsibility for Failures

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

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

In the case of a domain name failing DNSSEC validation, even when this is due to a misconfiguration of the domain, that is the sole responsibility of the domain owner.
Any assistance or mitigation responses undertaken by other parties to mitigate the misconfiguration of a domain name by a domain owner and/or administrator, especially operators of DNS recursive resolvers, are optional and at the pleasure of those parties. This can the use of a Negative Trust Anchor [RFC7646] and/or clearing the cache in particular DNS resolvers.

4. Comparison to Other DNS Misconfigurations

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

5. Other Considerations

5.1. Security Considerations

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

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

5.2. Privacy Considerations

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

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

There are no IANA considerations in this document.

6. Acknowledgements

- William Brown

7. References

7.1. Normative References


7.2. Informative References


Appendix A. Document Change Log

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

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

Individual-01: Fixed nits identified by William Brown

Individual-02: Updated prior to IETF-91

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

WG-01: Updated doc to keep it from expiring

WG-02: Removed RFC 2119 reference in XML

WG-03 to 04: Refreshed draft and broadened to all auth issues, not just DNSSEC.

Appendix B. Open Issues

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

Fix I-D xref
Author’s Address

Jason Livingood
Comcast

Email: jason_livingood@comcast.com
In Case of DNSSEC Validation Failures, Do Not Change Resolvers
draft-livingood-dnsop-dont-switch-resolvers-04

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress.”

This Internet-Draft will expire on August 22, 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.
1. Introduction

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

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

When these DNSSEC validation failures do occur, end users SHOULD NOT change to a non-validating DNS resolver, as that would downgrade their security. They should instead wait until the authoritative domain operator updates their DNS records to resolve the error and then for that change to propagate across the Internet’s DNS.
resolvers, the timing of which may be dependent upon the Time To Live (TTL) settings in the old and/or erroneous DNS resource records.

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

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

2. Reasons for DNSSEC Validation Failure

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

3. Misunderstanding DNSSEC Validation Failures

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

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

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

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

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

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

As a recommended best practice: In order to be as safe and secure as possible, end users SHOULD NOT change to DNS resolvers that do not perform DNSSEC validation as a workaround when DNSSEC validation failures occur.
Even if a website in a domain seems to look "normal" and valid, according to the DNSSEC protocol, that domain is not secure. Domains that fail DNSSEC validation may fail due to an actual security incident or compromise, and may be in control of hackers or there could be other significant security issues with the domain. Thus, switching to a non-validating resolver to restore access to a domain that fails DNSSEC validation is NOT recommended and is potentially harmful to end user security.

6. Other Considerations

6.1. Recommendations for Validating Resolver Operators

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

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

6.2. Security Considerations

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

In the case of a DNSSEC validation failure, if an end user changes to a non-validating resolver they can subject themselves to increased security risks and threats against which DNSSEC may have provided protection.
As a result, in order to protect their security, users SHOULD NOT switch to a non-validating resolver when a DNSSEC validation failure occurs.

6.3. Privacy Considerations

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

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

6.4. IANA Considerations

There are no IANA considerations in this document.

7. Acknowledgements

- William Brown
- Peter Koch

8. References

8.1. Normative References


8.2. Informative References

[I-D.livingood-dnsop-auth-dnssec-mistakes]
8.3. URIs


Appendix A. Document Change Log

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

Individual-00: First version published as an individual draft.
Individual-01: Fixed nits identified by William Brown
Individual-02: Updated prior to IETF-91
WG-00: Renamed at request of DNSOP co-chairs
WG-01: Updated doc to keep it from expiring
WG-02: Addressed some feedback from Peter Koch on RFC 2119 text, changed from BCP to Informational since this is more a recommended practice, added a section with recommendations for operators.
WG-03 to 04: Refreshed document

Appendix B. Open Issues

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

Fix I-D xref

Author’s Address

Jason Livingood
Comcast

Email: jason_livingood@comcast.com
Considerations for Large Authoritative DNS Servers Operators

draft-moura-dnsop-authoritative-recommendations-04

Abstract

This document summarizes recent research work exploring DNS configurations and offers specific, tangible considerations to operators for configuring authoritative servers.

This document is not an Internet Standards Track specification; it is published for informational purposes.

Ed note

This draft will be renamed to draft-moura-dnsop-large-authoritative-considerations in case adopted by the WG, to reflect the new title.

Text inside square brackets ([RF:ABC]) refers to:

- individual comments we have received about the draft, and enumerated under <https://github.com/gmmoura/draft-moura-dnsop-authoritative-recommendations/blob/master/reviews/reviews-dnsop.md>.

- Issues listed on our Github repository

Both types will be removed before publication.

This draft is being hosted on GitHub - <https://github.com/gmmoura/draft-moura-dnsop-authoritative-recommendations>, where the most recent version of the document and open issues can be found. 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.
Table of Contents

1. Introduction ................................. 3
2. Background ................................. 3
3. C1: Use equally strong IP anycast in every authoritative server (NS) for better load distribution ................................. 5
5. C3: Collecting Detailed Anycast Catchment Maps Ahead of Actual Deployment Can Improve Engineering Designs ................. 7
6. C4: When under stress, employ two strategies ................. 9
7. C5: Consider longer time-to-live values whenever possible .. 10
8. Security considerations .......................... 12
9. Privacy Considerations .......................... 12
10. IANA considerations .......................... 13
11. Acknowledgements ........................... 13
12. References ................................ 13
    12.1. Normative References ....................... 13
    12.2. Informative References .................... 14
Authors’ Addresses ............................ 16
1. Introduction

This document summarizes recent research work exploring DNS configurations and offers specific tangible considerations to DNS authoritative servers operators (DNS operators hereafter). [RF:JAb2], [RF:MSJ1], [RF:DW2]. The considerations (C1-C5) presented in this document are backed by previous research work, which used wide-scale Internet measurements upon which to draw their conclusions. This document describes the key engineering options, and points readers to the pertinent papers for details and [RF:Issue15] other research works related to each consideration here presented.

[RF:JAb1, Issue#2, SJa-02]. These considerations are designed for operators of "large" authoritative servers. In this context, "large" authoritative servers refers to those with a significant global user population, like TLDs, run by a single or multiple operators. These considerations may not be appropriate for smaller domains, such as those used by an organization with users in one city or region, where goals such as uniform low latency are less strict.

It is likely that these considerations might be useful in a wider context, such as for any stateless/short-duration, anycasted service. Because the conclusions of the studies don’t verify this fact, the wording in this document discusses DNS authoritative services only ([RF:Issue13]).

2. Background

The domain name system (DNS) has main two types of DNS servers: authoritative servers and recursive resolvers. Figure 1 shows their relationship. An authoritative server (ATn in Figure 1) knows the content of a DNS zone from local knowledge, and thus can answer queries about that zone without needing to query other servers [RFC2181]. A recursive resolver (Re_n) is a program that extracts information from name servers in response to client requests [RFC1034]. A client (stub in Figure 1) refers to stub resolver [RFC1034] that is typically located within the client software.
DNS queries/responses contribute to user’s perceived latency and affect user experience [Sigla2014], and the DNS system has been subject to repeated Denial of Service (DoS) attacks (for example, in November 2015 [Moura16b]) in order to degrade user experience.

To reduce latency and improve resiliency against DoS attacks, DNS uses several types of server 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 them referred to as anycast instance [RFC8499]) and (iii) by using 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 single optimal design.

In the next sections we cover specific considerations (C1-C5) for large authoritative DNS servers operators.
3. C1: Use equally strong IP anycast in every authoritative server (NS) for better load distribution

Authoritative DNS servers operators announce their authoritative servers as NS records[RFC1034]. Different authoritatives for a given zone should return the same content, typically by staying synchronized using DNS zone transfers (AXFR[RFC5936] and IXFR[RFC1995]) to coordinate the authoritative zone data to return to their clients.

DNS heavily relies upon replication to support high reliability, capacity and to reduce latency [Moura16b]. DNS has two complementary mechanisms to replicate the service. First, the protocol itself supports nameserver replication of DNS service for a DNS zone through the use of multiple nameservers that each operate on different IP addresses, listed by a zone’s NS records. Second, each of these network addresses can run from multiple physical locations through the use of IP anycast[RFC1546][RFC4786][RFC7094], by announcing the same IP address from each instance and allowing Internet routing (BGP[RFC4271]) to associate clients with their topologically nearest anycast instance. Outside the DNS protocol, replication can be achieved by deploying load balancers at each physical location. Nameserver replication is recommended for all zones (multiple NS records), and IP anycast is used by most large zones such as the DNS Root, most top-level domains[Moura16b] and large commercial enterprises, governments and other organizations.

Most DNS operators strive to reduce latency for users of their service. However, because they control only their authoritative servers, and not the recursive resolvers communicating with those servers, it is difficult to ensure that recursives will be served by the closest authoritative server. Server selection is up to the recursive resolver’s software implementation, and different software vendors and releases employ different criteria to chose which authoritative servers with which to communicate.

Knowing how recursives choose authoritative servers is a key step to better engineer the deployment of authoritative servers. [Mueller17b] evaluates this with a measurement study in which they deployed seven unicast authoritative name servers in different global locations and queried these authoritative servers from more than 9k RIPE Atlas probes and and their respective recursive resolvers.

In the wild, [Mueller17b] found that recursives query all available authoritative servers, regardless of the observed latency. But the distribution of queries tend 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 authoritative. These results were obtained by aggregating results from all vantage points and not specific to any vendor/version.

The hypothesis is that this behavior is a consequence of two main criteria employed by resolvers when choosing authoritatives: performance (lower latency) and diversity of authoritatives, where a resolver checks all authoritative servers to determine which is closer and to provide alternatives if one is unavailable.

For a DNS operator, this policy means that latency of all authoritatives (NS records [RF:SJa-01]) matter, so all must be similarly capable, since all available authoritatives will be queried by most recursive resolvers. Since unicast cannot deliver good latency worldwide (a unicast authoritative server in Europe will always have high latency to resolvers in California, for example, given its geographical distance), [Mueller17b] recommends to DNS operators that they deploy equally strong IP anycast in every authoritative server (i.e., on each NS record [RF:SJa-01]), in terms of number of instances and peering, and, consequently, to phase out unicast, so they can deliver good latency values to global clients. However, [Mueller17b] also notes that DNS operators should also take architectural considerations into account when planning for deploying anycast [RFC1546].

This consideration was deployed at the ".nl" TLD zone, which originally had seven authoritative servers (mixed unicast/anycast setup). .nl has moved in early 2018 to a setup with 4 anycast authoritative name servers. This is not to say that .nl was the first - other zones, have been running anycast only authoritatives (e.g., .be since 2013). [Mueller17b] contribution is to show that unicast cannot deliver good latency worldwide, and that anycast has to be deployed to deliver good latency worldwide.


A common metric when choosing an anycast DNS provider or setting up an anycast service is the number of anycast instances[RFC4786], i.e., the number of global locations from which the same address is announced with BGP. Intuitively, one could think that more instances will lead to shorter response times.

However, this is not necessarily true. In fact, [Schmidt17a] found that routing can matter more than the total number of locations. They analyzed the relationship between the number of anycast instances and the performance of a service (latency-wise, RTT) and measured the overall performance of four DNS Root servers, namely C, F, K and L, from more than 7.9k RIPE Atlas probes.
[Schmidt17a] found that C-Root, a smaller anycast deployment consisting of only 8 instances (they refer to anycast instance as anycast site), provided a very similar overall performance than that of the much larger deployments of K and L, with 33 and 144 instances respectively. The median RTT for C, K and L Root was between 30-32ms.

Given that Atlas has better coverage in Europe than other regions, the authors specifically analyzed results per region and per country (Figure 5 in [Schmidt17a]), and show that Atlas bias to Europe does not change the conclusion that location of anycast instances dominates latency. [RF:Issue12]

[Schmidt17a] consideration for DNS operators when engineering anycast services is consider factors other than just the number of instances (such as local routing connectivity) when designing for performance. They showed that 12 instances can provide reasonable latency, given they are globally distributed and have good local interconnectivity. However, more instances can be useful for other reasons, such as when handling DDoS attacks [Moura16b].

5. C3: Collecting Detailed Anycast Catchment Maps Ahead of Actual Deployment Can Improve Engineering Designs

An anycast DNS service may have several dozens or even more than one hundred instances (such as L-Root does). Anycast leverages Internet routing to distribute the incoming queries to a service’s distributed anycast instances; in theory, BGP (the Internet’s defacto routing protocol) forwards incoming queries to a nearby anycast instance (in terms of BGP distance). However, usually queries are not evenly distributed across all anycast instances, as found in the case of L-Root [IcannHedge18].

Adding new instances to an anycast service may change the load distribution across all instances, leading to suboptimal usage of the service or even stressing some instances while others remain underutilized. This is a scenario that operators constantly face when expanding an anycast service. Besides, when setting up a new anycast service instance, operators cannot directly estimate the query distribution among the instances in advance of enabling the new instance.

To estimate the query loads across instances of an expanding service or a when setting up an entirely new service, operators need detailed anycast maps and catchment estimates (i.e., operators need to know which prefixes will be matched to which anycast instance). To do that, [Vries17b] developed a new technique enabling operators to carry out active measurements, using an open-source tool called...
Verfploeter (available at [VerfSrc]). Verfploeter maps a large portion of the IPv4 address space, allowing DNS operators to predict both query distribution and clients catchment before deploying new anycast instances.

[Vries17b] shows how this technique was used to predict both the catchment and query load distribution for the new anycast service of B-Root. Using two anycast instances in Miami (MIA) and Los Angeles (LAX) from the operational B-Root server, they sent ICMP echo packets to IP addresses to each IPv4 /24 on the Internet using a source address within the anycast prefix. Then, they recorded which instance the ICMP echo replies arrived at based on the Internet’s BGP routing. This analysis resulted in an Internet wide catchment map. Weighting was then applied to the incoming traffic prefixes based on of 1 day of B-Root traffic (2017-04-12, DITL datasets [Ditl17]). The combination of the created catchment mapping and the load per prefix created an estimate predicting that 81.6% of the traffic would go to the LAX instance. The actual value was 81.4% of traffic going to LAX, showing that the estimation was pretty close and the Verfploeter technique was a excellent method of predicting traffic loads in advance of a new anycast instance deployment ([Vries17b] also uses the term anycast site to refer to anycast instance).

Besides that, Verfploeter can also be used to estimate how traffic shifts among instances when BGP manipulations are executed, such as AS Path prepending that is frequently used by production networks during DDoS attacks. A new catchment mapping for each prepending configuration configuration: no prepending, and prepending with 1, 2 or 3 hops at each instance. Then, [Vries17b] shows that this mapping can accurately estimate the load distribution for each configuration.

An important operational takeaway from [Vries17b] is that DNS operators can make informed choices when engineering new anycast instances or when expending new ones by carrying out active measurements using Verfploeter in advance of operationally enabling the fully anycast service. Operators can spot sub-optimal routing situations early, with a fine granularity, and with significantly better coverage than using traditional measurement platforms such as RIPE Atlas.

To date, Verfploeter has been deployed on B-Root[Vries17b], on a operational testbed (Anycast testbed) [AnyTest], and on a large unnamed operator.

The consideration is therefore to deploy a small test Verfploeter-enabled platform in advance at a potential anycast instance may reveal the realizable benefits of using that instance as an anycast interest, potentially saving significant financial and labor costs of
deploying hardware to a new instance that was less effective than as had been hoped.

6. C4: When under stress, employ two strategies

DDoS attacks are becoming bigger, cheaper, and more frequent [Moura16b]. The most powerful recorded DDoS attack to DNS servers to date reached 1.2 Tbps, by using IoT devices [Perlroth16]. Such attacks call for an answer for the following question: how should a DNS operator engineer its anycast authoritative DNS server react to the stress of a DDoS attack? This question is investigated in study [Moura16b] in which empirical observations are grounded with the following theoretical evaluation of options.

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 [Moura16b]. 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 under stress, overloaded with incoming traffic, has two options:

- It can withdraw or pre-prepend its route to some or to all of its neighbors, ([RF:Issue3]) perform other traffic shifting tricks (such as reducing the propagation of its announcements using BGP communities[RFC1997]) which shrinks portions of its catchment), use FlowSpec [RFC5575] or other upstream communication mechanisms to deploy upstream filtering. The goals of these techniques is to perform some combination of shifting of both legitimate and attack traffic to other anycast instances (with hopefully greater capacity) or to block the traffic entirely.

- Alternatively, it can become a degraded absorber, continuing to operate, but with overloaded ingress routers, dropping some incoming legitimate requests due to queue overflow. However, continued operation will also absorb traffic from attackers in its catchment, protecting the other anycast instances.
[Moura16b] saw both of these behaviors in practice in the Root DNS events, observed through instance reachability and route-trip time (RTTs). These options represent different uses of an anycast deployment. The withdrawal strategy causes anycast to respond as a waterbed, with stress displacing queries from one instance to others. The absorption strategy behaves as a conventional mattress, compressing under load, with some queries getting delayed or dropped. Although described as strategies and policies, these outcomes are the result of several factors: the combination of operator and host ISP routing policies, routing implementations withdrawing under load, the nature of the attack, and the locations of the instances and the attackers. Some policies are explicit, such as the choice of local-only anycast instances, or operators removing an instance for maintenance or modifying routing to manage load. However, under stress, the choices of withdrawal and absorption can also be results that emerge from a mix of explicit choices and implementation details, such as BGP timeout values.

[Moura16b] speculates that more careful, explicit, and automated management of policies may provide stronger defenses to overload, an area currently under study. For DNS operators, that means that besides traditional filtering, two other options are available (withdraw/prepend/communities or isolate instances), and the best choice depends on the specifics of the attack.

Note that this consideration refers to the operation of one anycast service, i.e., one anycast NS record. However, DNS zones with multiple NS anycast services may expect load to spill from one anycast server to another, as resolvers switch from authoritative to authoritative when attempting to resolve a name [Mueller17b].

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

[RF:Issue7]: this section has been completely rewritten.

Caching is the cornerstone of good DNS performance and reliability. A 15 ms response to a new DNS query is fast, but a 1 ms cache hit to a repeat query is far faster. Caching also protects users from short outages and can mute even significant DDoS attacks [Moura18b].

DNS record TTLs (time-to-live values) directly control cache durations [RFC1034][RFC1035] and, therefore, 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 examined their interaction with DNS [Moura18b], but no research provides considerations about what TTL values are good. With this
goal Moura et. al. [Moura19a] carried out a measurement study investigating TTL choices and its impact on user experience.

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

- Longer caching results in faster responses, given that cache hits are faster than cache misses in resolvers. [Moura19a] shows that the change in TTL for .uy TLD from 1 day to 5 minutes reduced the RTT from 15k Atlas vantage points significantly: the median was reduced from 28.7ms to 8ms, while the 75%ile decreased from 183ms to 21ms.

- Longer caching results in lower DNS traffic: authoritative servers will experience less traffic if TTLs are extended, given that repeated queries will be answered by resolver caches.

- Longer caching results in lower cost if DNS is metered: some DNS-As-A-Service providers charges are metered, with a per query cost (often added to a fixed monthly cost).

- Longer caching is more robust to DDoS attacks on DNS: DDoS attacks on a DNS service provider harmed several prominent websites [Perlroth16]. Recent work has shown that DNS caching can greatly reduce the effects of DDoS on DNS, provided caches last longer than the attack [Moura18b].

- Shorter caching supports operational changes: An easy way to transition from an old server to a new one is to change the DNS records. Since there is no method to remove cached DNS records, the TTL duration represents a necessary transition delay to fully shift to a new server, so low TTLs allow more rapid transition. However, when deployments are planned in advance (that is, longer than the TTL), then TTLs can be lowered ‘’just-before’’ a major operational change, and raised again once accomplished.

- Shorter caching can with DNS-based load balancing: Some DDoS-scrubbing services use 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 be ready to respond to a potential attack.

As such, choice of TTL depends in part on external factors so no single recommendation is appropriate for all. Organizations must weigh these trade-offs to find a good balance. Still, some guidelines can be used when choosing TTLs:
For general users, [Moura19a] recommends longer TTLs, of at least one hour, and ideally 4, 8, 12, or 24 hours. Assuming planned maintenance can be scheduled at least a day in advance, long TTLs have little cost.

For TLD operators: TLD operators that allow public registration of domains (such as most ccTLDs and .com, .net, .org) host, in their zone files, NS records (and glues if in-bailiwick) of their respective domains. [Moura19a] shows that most resolvers will use TTL values provided by the child delegations, but some will choose the TTL provided by the parents. As such, similarly to general users, [Moura19a] recommends longer TTLs for NS records of their delegations (at least one hour, preferably more).

Users of DNS-based load balancing or DDoS-prevention may require short TTLs: TTLs may be as short as 5 minutes, although 15 minutes may provide sufficient agility for many operators. Shorter TTLs here help agility; they are are an exception to the consideration for longer TTLs.

Use A/AAAA and NS records: TTLs of A/AAAA records should be shorter or equal to the TTL for NS records for in-bailiwick authoritative DNS servers, given that the authors [Moura19a] found that, for such scenarios, once NS record expires, their associated A/AAAA will also be updated (glue is sent by the parents). For out-of-bailiwick servers, A and NS records are usually cached independently, so different TTLs, if desired, will be effective. In either case, short A and AAAA records may be desired if DDoS-mitigation services are an option.

8. Security considerations

This document suggests the use of [I-D.ietf-dnsop-serve-stale]. It be noted that usage of such methods may affect data integrity of DNS information. This document describes methods of mitigating changes of a denial of service threat within a DNS service.

As this document discusses research, there are no further security considerations, other than the ones mentioned in the normative references.

9. Privacy Considerations

This document does not add any practical new privacy issues.
10. IANA considerations

This document has no IANA actions.

11. Acknowledgements

This document is a summary of the main considerations of six research works referred in this document. As such, they were only possible thanks to the hard work of the authors of these research works.

The authors of this document are also co-authors of these research works. However, not all thirteen authors of these research papers are also authors of this document. We would like to thank those not included in this document’s author list for their work: Ricardo de O. Schmidt, Wouter B de Vries, Moritz Mueller, Lan Wei, Cristian Hesselman, Jan Harm Kuipers, Pieter-Tjerk de Boer and Aiko Pras.

We would like also to thank the various reviewers of different versions of this draft: Duane Wessels, Joe Abley, Toema Gavrichenko, John Levine, Michael StJohns, Kristof Tuytelaers, Stefan Ubbink, Klaus Darilion and Samir Jafferali, and comments provided at the IETF DNSOP session (IETF104).

Besides those, we would like thank those who have been individually thanked in each research work, 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.

12. References

12.1. Normative References

[I-D.ietf-dnsop-serve-stale]


12.2. Informative References


Authors’ Addresses

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

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

Wes Hardaker
USC/Information Sciences Institute
PO Box 382
Davis 95617-0382
U.S.A.

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

John Heidemann
USC/Information Sciences Institute
4676 Admiralty Way
Marina Del Rey 90292-6695
U.S.A.

Phone: +1 (310) 448-8708
Email: johnh@isi.edu
Guidelines for Use of the Special Use Names Registry
draft-stw-6761ext-01

Abstract

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

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

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

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

However, there are some risks when the protocol designer attempts to re-use domain names and DNS, even (or especially) with modifications, to support a specific use case or protocol design or deployment constraint. These have been touched upon in several RFCs, and in the evolution of DNS protocol itself and the use of domain names as new needs and constraints appear. See in particular RFC 6055 ("IAB Thoughts on Encodings for Internationalized Domain Names"), RFC 6950 ("Architectural Considerations on Application Features in the DNS"),
and RFC 6943 ("Issues in Identifier Comparison for Security Purposes").

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

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

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

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

Trying to specify how special use domain names interact with the DNS is both necessary for interoperability and helpful in thinking through the proposed "special use". So a proponent of a special use name might discover, in the course of specifying the "special use" for the SUNR, that domain names will not meet the constraints at hand. But even if domain names seem like a good fit for the problem, there’s also no guidance in RFC 6761 to deciding what names might or might not be appropriate for the particular need.
The broader discussion of the general applicability of domain names to new needs is useful to consider, and owes a great deal to the RFCs already mentioned, especially RFC 6950, which "provides guidance to application designers and application protocol designers looking to use the DNS to support features in their applications." The consideration there of how to structure domain names and associated data is invaluable. For a different, and sometimes more comprehensive, view on some of the accumulated stresses on the DNS design, see also RFC 8324 ("DNS Privacy, Authorization, Special Uses, Encoding, Characters, Matching, and Root Structure: Time for Another Look?")

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

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

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

2. Current SUNR Use Cases

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

1. Proponents wish to reserve a name to serve a specific purpose in an IETF protocol, discussed as part of protocol definition in an IETF working group. Resolution of the name may be intended for a
limited scope (homenet) or outside of the DNS altogether (mDNS, DNSSD)

2. Proponents wish to reserve a name as used in a protocol developed outside of the IETF, in order to avoid potential collisions with other uses of the namespace. Possible sources of such collisions include future IETF protocols or ICANN’s policies for delegation of top-level domains (.onion, RFC 7686)

3. Proponents wish to reserve a name from any use in the public DNS, in order to support interoperability and avoid collision or abuse ("localhost," or draft-chapin-additional-reserved-tlds)

3. Guidelines for Special Use Name registration

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

1. Location of a name in the namespace is a consideration. A single-label name or "top level domain" can be attractive at first glance: they can be short and human-friendly, and there’s no obvious need to coordinate the use of a top-level label with a TLD operator by, for example, purchasing the use of a second-level domain such as example.org. But the reservation of a TLD also poses a unique challenge, whether the proponent is asking for it to be reserved from use in the DNS root zone, or asking for it to be added to the root zone: the IETF administers the SUNR, but does not control the root zone. Under RFC 2860, ICANN has that authority. More discussion on this point can be seen below, but as a practical matter, IETF Working Groups should not make such requests without compelling justification, and the IESG should not advance them without asking what other options might be available to satisfy relevant technical requirements. (Case: home.arpa, RFC 8375)

1. Compatibility with an installed base might be a compelling need to reserve a specific string as a single label or TLD. This does impose a burden of coordination on ICANN, the IETF, and the IAB, and adds to possible confusion for developers and operators across the wider internet, so the bar to proceeding in this way should be high. There should be significant benefit to interoperability, at the very least. (Case: .onion, .bit, etc.)

2. Preventing ICANN from delegating a name is not, by itself, a compelling reason to reserve it in the SUNR. There’s no written policy or agreement that says it would work, and
ICANN may have no process or policy under which it could
determine whether such a reservation should be granted. Risking name collision under different policies from
different authorities seems unwise, but so does using
standards action in one body to constrain policy in another.
(Case: home/corp/mail, draft-chapin-additional-tlds)

2. For names reserved as part of an IETF protocol, in a standards-
track RFC coming out of an IETF WG, proponents should consider
using .arpa (see the IAB note on home.arpa, and RFC 3172). This
can work whether the name is supposed to be instantiated in the
DNS or not, since the IAB sets policy for .arpa. (Case: home.arpa)

3. Reserved domain names that aren’t TLDs require less work for the
community because they don’t have to be coordinated with another
body. All such names, however, should be carefully considered
regarding the characteristics discussed above: do they need to
exist in the public DNS, or just be valid in a limited scope, or
be reserved for another protocol? do they have semantic meaning
outside of the specific protocol or scope? do they need to be
human-friendly? etc. This may require adding some new questions
to the RFC 6761 list, which talks about how the names are treated
by DNS but otherwise not much about why they’re being reserved or
how they’re being used. (Case: home.arpa)

4. For names initially reserved or used outside of the IETF, for
which a proponent wants to add a special use name registry entry,
the bar should be just as high. For single labels in particular,
the IESG and the community should require both a stable
specification and some assurance that a one-time delegation won’t
multiply as the protocol evolves or the community forks. This
may require a standards-track update to RFC 6761.

4. The Special Case of Top-Level Domains

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

As noted above, all of the cases described in this document are more
difficult if the proponents are attempting to reserve a single label
domain name, or "TLD". This is because the IETF delegated authority
some time ago to ICANN for the contents of the root zone of the DNS
(see RFC 2860).
RFC 6761 claims that the SUNR is based on a "protocol rule" with unchallengeable precedence over ICANN policy. However, it’s not clear exactly what this means in practice. There’s no process for making a request to ICANN to add a TLD to the root zone, or a string to the list of names ICANN commits won’t be delegated, and it seems likely that the effort of inventing one and coordinating it with ICANN would not be justified unless there was a compelling need that couldn’t be met any other way.

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

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

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

For these reasons, the bar for the IESG and the IETF community to agree to request a TLD in the SUNR-- either that it should never be delegated, or that it should be delegated according to conditions set by the IETF-- should be very high indeed. The IESG SHOULD NOT make
such requests without a compelling reason that cannot, as a matter of technical necessity, be met by a special use name elsewhere in the domain name space.

5. Acknowledgements

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

6. Informative References


Internet-Draft  Name Registration Considerations March 2019


Author’s Address

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

Email: suzworldwide@gmail.com
Algorithms for Domain Name System (DNS) Cookies construction
draft-sury-toorop-dns-cookies-algorithms-00

Abstract

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

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

This document updates [RFC7873]

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2019.
1. Introduction

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

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

In Section Section 2 instructions for constructing a Client Cookie are given

In Section Section 3 instructions for constructing a Server Cookie are given

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

1.2. Definitions

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

2. Constructing a Client Cookie

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

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

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

where "|" indicates concatenation.

3. Constructing a Server Cookie

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

The Server Cookie is not required to be changed periodically if a secure pseudorandom function is used.
The 128-bit Server Cookie consists of Sub-Fields: a 1 octet Version Sub-Field, a 1 octet Cookie Algorithm Sub-Field, a 2 octet Reserved Sub-Field, a 4 octet Timestamp Sub-Field and a 8 octet Hash Sub-Field.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |  Cookie Algo  |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Hash                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.1. The Version Sub-Field

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

3.2. The Cookie algo Sub-Field

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

3.3. The Reserved Sub-Field

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

3.4. The Timestamp Sub-Field

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

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

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

\[
\text{Hash} = \text{Cookie\_Algorithm}(\text{Client Cookie | Version | Cookie Algo | Reserved | TimeStamp, Server Secret})
\]

4. Cookie Algorithms

Implementation recommendations for Cookie Algorithms [DNSCOOKIE-IANA]:

<table>
<thead>
<tr>
<th>Number</th>
<th>Mnemonics</th>
<th>Client Cookie</th>
<th>Server Cookie</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FNV</td>
<td>MUST NOT</td>
<td>MUST NOT</td>
</tr>
<tr>
<td>2</td>
<td>HMAC-SHA-256-64</td>
<td>MUST NOT</td>
<td>MUST NOT</td>
</tr>
<tr>
<td>3</td>
<td>AES</td>
<td>MAY</td>
<td>MAY</td>
</tr>
<tr>
<td>4</td>
<td>SipHash24</td>
<td>MUST</td>
<td>MUST</td>
</tr>
</tbody>
</table>

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

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

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

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

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

6. References

6.1. Normative References


6.2. Informative References


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