

Network Working Group
Internet-Draft
Obsoletes: 7816 (if approved)
Intended status: Standards Track
Expires: January 18, 2019

S. Bortzmeyer
AFNIC
P. Hoffman
ICANN
July 17, 2018

DNS Query Name Minimisation to Improve Privacy
draft-bortzmeyer-rfc7816bis-00

Abstract

This document describes a technique to improve DNS privacy, a technique called "QNAME minimisation", where the DNS resolver no longer sends the full original QNAME to the upstream name server.

RFC EDITOR: PLEASE REMOVE BEFORE PUBLICATION. The original [RFC7816] had the experimental status. This document is intended for the standards track. It should be discussed in the IETF DNSOP (DNS Operations) Working Group, through its mailing list. The source of the document, as well as a list of open issues, is currently kept at Framagit [1].

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 January 18, 2019.

Copyright Notice

Copyright (c) 2018 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 and Background	2
2. QNAME Minimisation	3
3. Possible Issues	4
4. Protocol and Compatibility Discussion	6
5. Operational Considerations	6
6. Performance Considerations	7
7. Results of the Experimentation	7
8. Security Considerations	8
9. Implementation status - RFC EDITOR: PLEASE REMOVE BEFORE PUBLICATION	8
10. References	9
10.1. Normative References	9
10.2. Informative References	9
10.3. URIs	10
Appendix A. An Algorithm to Perform QNAME Minimisation	10
Appendix B. Alternatives	11
Acknowledgments	12
Changes from RFC 7816	12
Authors' Addresses	12

1. Introduction and Background

The problem statement is described in [I-D.bortzmeyer-dprive-rfc7626-bis]. The terminology ("QNAME", "resolver", etc.) is defined in [I-D.ietf-dnsop-terminology-bis]. This specific solution is not intended to fully solve the DNS privacy problem; instead, it should be viewed as one tool amongst many.

QNAME minimisation follows the principle explained in Section 6.1 of [RFC6973]: the less data you send out, the fewer privacy problems you have.

Before QNAME minimisation, when a resolver received the query "What is the AAAA record for www.example.com?", it sent to the root (assuming a cold resolver, whose cache is empty) the very same question. Sending the full QNAME to the authoritative name server was a tradition, not a protocol requirement. In a conversation with the author in January 2015, Paul Mockapetris explained that this tradition comes from a desire to optimise the number of requests,

when the same name server is authoritative for many zones in a given name (something that was more common in the old days, where the same name servers served .com and the root) or when the same name server is both recursive and authoritative (something that is strongly discouraged now). Whatever the merits of this choice at this time, the DNS is quite different now.

2. QNAME Minimisation

The idea is to minimise the amount of data sent from the DNS resolver to the authoritative name server. In the example in the previous section, sending "What are the NS records for .com?" would have been sufficient (since it will be the answer from the root anyway). The rest of this section describes the recommended way to do QNAME minimisation -- the way that maximises privacy benefits (other alternatives are discussed in the appendices).

Instead of sending the full QNAME and the original QTYPE upstream, a resolver that implements QNAME minimisation and does not already have the answer in its cache sends a request to the name server authoritative for the closest known ancestor of the original QNAME. The request is done with:

- o the QTYPE NS
- o the QNAME that is the original QNAME, stripped to just one label more than the zone for which the server is authoritative

For example, a resolver receives a request to resolve foo.bar.baz.example. Let's assume that it already knows that ns1.nic.example is authoritative for .example and the resolver does not know a more specific authoritative name server. It will send the query QTYPE=NS,QNAME=baz.example to ns1.nic.example.

The minimising resolver works perfectly when it knows the zone cut (zone cuts are described in Section 6 of [RFC2181]). But zone cuts do not necessarily exist at every label boundary. If we take the name www.foo.bar.example, it is possible that there is a zone cut between "foo" and "bar" but not between "bar" and "example". So, assuming that the resolver already knows the name servers of .example, when it receives the query "What is the AAAA record of www.foo.bar.example?", it does not always know where the zone cut will be. To find the zone cut, it will query the .example name servers for the NS records for bar.example. It will get a NODATA response, indicating that there is no zone cut at that point, so it has to query the .example name servers again with one more label, and so on. (Appendix A describes this algorithm in deeper detail.)

Here are more detailed examples of queries with QNAME minimisation:

www.isc.org, cold cache, aggressive algorithm:

QTYPE	QNAME	TARGET	NOTE
NS	org	root nameserver	
NS	isc.org	Afilias nameserver	
NS	www.isc.org	ISC nameserver	"www" may be delegated
A	www.isc.org	ISC nameserver	

www.isc.org, cold cache, lazy algorithm (for a cold cache, it is the same algorithm as now):

QTYPE	QNAME	TARGET	NOTE
A	www.isc.org	root nameserver	
A	www.isc.org	Afilias nameserver	
A	www.isc.org	ISC nameserver	

www.isc.org, warm cache (all NS RRsets are known), both algorithms:

QTYPE	QNAME	TARGET	NOTE
A	www.isc.org	ISC nameserver	

www.example.org, warm cache (but for isc.org only, example.org's NS RRset is not known), aggressive algorithm

QTYPE	QNAME	TARGET	NOTE
NS	example.org	Afilias nameserver	
NS	www.example.org	Example nameserver	
A	www.example.org	Example nameserver	

Since the information about the zone cuts will be stored in the resolver's cache, the performance cost is probably reasonable. Section 6 discusses this performance discrepancy further.

Note that DNSSEC-validating resolvers already have access to this information, since they have to know the zone cut (the DNSKEY record set is just below; the DS record set is just above).

3. Possible Issues

TODO may be remove the whole section now that it is no longer experimental?

QNAME minimisation is legal, since the original DNS RFCs do not mandate sending the full QNAME. So, in theory, it should work without any problems. However, in practice, some problems may occur

(see [Huque-QNAME-Min] for an analysis and [Huque-QNAME-storify] for an interesting discussion on this topic).

Some broken name servers do not react properly to QTYPE=NS requests. For instance, some authoritative name servers embedded in load balancers reply properly to A queries but send REFUSED to NS queries. This behaviour is a protocol violation, and there is no need to stop improving the DNS because of such behaviour. However, QNAME minimisation may still work with such domains, since they are only leaf domains (no need to send them NS requests). Such a setup breaks more than just QNAME minimisation. It breaks negative answers, since the servers don't return the correct SOA, and it also breaks anything dependent upon NS and SOA records existing at the top of the zone.

Another way to deal with such incorrect name servers would be to try with QTYPE=A requests (A being chosen because it is the most common and hence a QTYPE that will always be accepted, while a QTYPE NS may ruffle the feathers of some middleboxes). Instead of querying name servers with a query "NS example.com", we could use "A *.example.com" and see if we get a referral. TODO this is what Unbound does

A problem can also appear when a name server does not react properly to ENTs (Empty Non-Terminals). If ent.example.com has no resource records but foobar.ent.example.com does, then ent.example.com is an ENT. Whatever the QTYPE, a query for ent.example.com must return NODATA (NOERROR / ANSWER: 0). However, some name servers incorrectly return NXDOMAIN for ENTs. If a resolver queries only foobar.ent.example.com, everything will be OK, but if it implements QNAME minimisation, it may query ent.example.com and get an NXDOMAIN. See also Section 3 of [DNS-Res-Improve] for the other bad consequences of this bad behaviour.

A possible solution, currently implemented in Knot or Unbound, is to retry with the full query when you receive an NXDOMAIN. It works, but it is not ideal for privacy.

Other practices that do not conform to the DNS protocol standards may pose a problem: there is a common DNS trick used by some web hosters that also do DNS hosting that exploits the fact that the DNS protocol (pre-DNSSEC) allows certain serious misconfigurations, such as parent and child zones disagreeing on the location of a zone cut. Basically, they have a single zone with wildcards for each TLD, like:

```
*.example.          60  IN  A    192.0.2.6
```

(They could just wildcard all of ".*", which would be sufficient. We don't know why they don't do it.)

This lets them have many web-hosting customers without having to configure thousands of individual zones on their name servers. They just tell the prospective customer to point their NS records at the hoster's name servers, and the web hoster doesn't have to provision anything in order to make the customer's domain resolve. NS queries to the hoster will therefore not give the right result, which may endanger QNAME minimisation (it will be a problem for DNSSEC, too).

4. Protocol and Compatibility Discussion

QNAME minimisation is compatible with the current DNS system and therefore can easily be deployed; since it is a unilateral change to the resolver, it does not change the protocol. (Because it is a unilateral change, resolver implementers may do QNAME minimisation in slightly different ways; see the appendices for examples.)

One should note that the behaviour suggested here (minimising the amount of data sent in QNAMEs from the resolver) is NOT forbidden by Section 5.3.3 of [RFC1034] or Section 7.2 of [RFC1035]. As stated in Section 1, the current method, sending the full QNAME, is not mandated by the DNS protocol.

One may notice that many documents that explain the DNS and that are intended for a wide audience incorrectly describe the resolution process as using QNAME minimisation (e.g., by showing a request going to the root, with just the TLD in the query). As a result, these documents may confuse readers that use them for privacy analysis.

5. Operational Considerations

TODO what to do if the resolver forwards? Unbound disables QNAME minimisation in that case, since the forwarder will see everything, anyway. What should a minimising resolver do when forwarding the request to a forwarder, not to an authoritative name server? Send the full qname? Minimises? (But how since we do not know the zone cut?)

The administrators of the forwarders, and of the authoritative name servers, will get less data, which will reduce the utility of the statistics they can produce (such as the percentage of the various QTYPES).

DNS administrators are reminded that the data on DNS requests that they store may have legal consequences, depending on your jurisdiction (check with your local lawyer).

6. Performance Considerations

The main goal of QNAME minimisation is to improve privacy by sending less data. However, it may have other advantages. For instance, if a root name server receives a query from some resolver for A.example followed by B.example followed by C.example, the result will be three NXDOMAINs, since .example does not exist in the root zone. Under query name minimisation, the root name servers would hear only one question (for .example itself) to which they could answer NXDOMAIN, thus opening up a negative caching opportunity in which the full resolver could know a priori that neither B.example nor C.example could exist. Thus, in this common case the total number of upstream queries under QNAME minimisation would be counterintuitively less than the number of queries under the traditional iteration (as described in the DNS standard). TODO mention [RFC8020]?

QNAME minimisation may also improve lookup performance for TLD operators. For a typical TLD, delegation-only, and with delegations just under the TLD, a two-label QNAME query is optimal for finding the delegation owner name.

QNAME minimisation can decrease performance in some cases -- for instance, for a deep domain name (like www.host.group.department.example.com, where host.group.department.example.com is hosted on example.com's name servers). Let's assume a resolver that knows only the name servers of example.com. Without QNAME minimisation, it would send these example.com name servers a query for www.host.group.department.example.com and immediately get a specific referral or an answer, without the need for more queries to probe for the zone cut. For such a name, a cold resolver with QNAME minimisation will, depending on how QNAME minimisation is implemented, send more queries, one per label. Once the cache is warm, there will be no difference with a traditional resolver. Actual testing is described in [Huque-QNAME-Min]. Such deep domains are especially common under ip6.arpa.

7. Results of the Experimentation

TODO various experiences from actual deployments, problems heard.
TODO the Knot bug #339 <https://gitlab.labs.nic.cz/knot/knot-resolver/issues/339?> TODO Problems with AWS <https://forums.aws.amazon.com/thread.jspa?threadID=269116?>

8. Security Considerations

QNAME minimisation's benefits are clear in the case where you want to decrease exposure to the authoritative name server. But minimising the amount of data sent also, in part, addresses the case of a wire sniffer as well as the case of privacy invasion by the servers. (Encryption is of course a better defense against wire sniffers, but, unlike QNAME minimisation, it changes the protocol and cannot be deployed unilaterally. Also, the effect of QNAME minimisation on wire sniffers depends on whether the sniffer is on the DNS path.)

QNAME minimisation offers zero protection against the recursive resolver, which still sees the full request coming from the stub resolver.

All the alternatives mentioned in Appendix B decrease privacy in the hope of improving performance. They must not be used if you want maximum privacy.

9. Implementation status - RFC EDITOR: PLEASE REMOVE BEFORE PUBLICATION

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Unbound has QNAME minimisation for several years, and it is now the default. It has two modes, strict (no workaround for broken authoritative name servers) and "lax" (retries when there is a NXDOMAIN). TODO Ralph Dolmans talk at OARC https://indico.dns-oarc.net/event/22/contributions/332/attachments/310/542/unbound_qnamemin_oarc24.pdf

Knot resolver also has QNAME minimisation since 2016, and it is activated by default.

BIND has QNAME minimisation since BIND 9.13.2, released in July 2018. Like Unbound, it has several modes, with or without workarounds for broken authoritative name servers.

PowerDNS does not have QNAME minimisation. TODO
<https://github.com/PowerDNS/pdns/issues/2311>

The public DNS resolver at Cloudflare ("1.1.1.1") has QNAME minimisation (it uses Knot).

10. References

10.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.

10.2. Informative References

- [DNS-Res-Improve] Vixie, P., Joffe, R., and F. Neves, "Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness", Work in Progress, draft-vixie-dnsext-resimprove-00, June 2010.
- [HAMMER] Kumari, W., Arends, R., Woolf, S., and D. Migault, "Highly Automated Method for Maintaining Expiring Records", Work in Progress, draft-wkumari-dnsop-hammer-01, July 2014.
- [Huque-QNAME-Min] Huque, S., "Query name minimization and authoritative server behavior", May 2015, <<https://indico.dns-oarc.net/event/21/contribution/9>>.

- [Huque-QNAME-storify]
Huque, S., "Qname Minimization @ DNS-OARC", May 2015,
<<https://storify.com/shuque/qname-minimization-dns-oarc>>.
- [I-D.bortzmeyer-dprive-rfc7626-bis]
Bortzmeyer, S. and S. Dickinson, "DNS Privacy
Considerations", draft-bortzmeyer-dprive-rfc7626-bis-01
(work in progress), July 2018.
- [I-D.ietf-dnsop-terminology-bis]
Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS
Terminology", draft-ietf-dnsop-terminology-bis-11 (work in
progress), July 2018.
- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS
Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997,
<<https://www.rfc-editor.org/info/rfc2181>>.
- [RFC7816] Bortzmeyer, S., "DNS Query Name Minimisation to Improve
Privacy", RFC 7816, DOI 10.17487/RFC7816, March 2016,
<<https://www.rfc-editor.org/info/rfc7816>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running
Code: The Implementation Status Section", BCP 205,
RFC 7942, DOI 10.17487/RFC7942, July 2016,
<<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8020] Bortzmeyer, S. and S. Huque, "NXDOMAIN: There Really Is
Nothing Underneath", RFC 8020, DOI 10.17487/RFC8020,
November 2016, <<https://www.rfc-editor.org/info/rfc8020>>.

10.3. URIs

- [1] <https://framagit.org/bortzmeyer/rfc7816-bis>

Appendix A. An Algorithm to Perform QNAME Minimisation

This algorithm performs name resolution with QNAME minimisation in the presence of zone cuts that are not yet known.

Although a validating resolver already has the logic to find the zone cuts, implementers of other resolvers may want to use this algorithm to locate the cuts. This is just a possible aid for implementers; it is not intended to be normative:

- (0) If the query can be answered from the cache, do so; otherwise, iterate as follows:

- (1) Find the closest enclosing NS RRset in your cache. The owner of this NS RRset will be a suffix of the QNAME -- the longest suffix of any NS RRset in the cache. Call this ANCESTOR.
- (2) Initialise CHILD to the same as ANCESTOR.
- (3) If CHILD is the same as the QNAME, resolve the original query using ANCESTOR's name servers, and finish.
- (4) Otherwise, add a label from the QNAME to the start of CHILD.
- (5) If you have a negative cache entry for the NS RRset at CHILD, go back to step 3.
- (6) Query for CHILD IN NS using ANCESTOR's name servers. The response can be:
 - (6a) A referral. Cache the NS RRset from the authority section, and go back to step 1.
 - (6b) An authoritative answer. Cache the NS RRset from the answer section, and go back to step 1.
 - (6c) An NXDOMAIN answer. Return an NXDOMAIN answer in response to the original query, and stop.
 - (6d) A NOERROR/NODATA answer. Cache this negative answer, and go back to step 3.

Appendix B. Alternatives

Remember that QNAME minimisation is unilateral, so a resolver is not forced to implement it exactly as described here.

There are several ways to perform QNAME minimisation. See Section 2 for the suggested way. It can be called the aggressive algorithm, since the resolver only sends NS queries as long as it does not know the zone cuts. This is the safest, from a privacy point of view. Another possible algorithm, not fully studied at this time, could be to "piggyback" on the traditional resolution code. At startup, it sends traditional full QNAMEs and learns the zone cuts from the referrals received, then switches to NS queries asking only for the minimum domain name. This leaks more data but could require fewer changes in the existing resolver codebase.

In the above specification, the original QTYPE is replaced by NS (or may be A, if too many servers react incorrectly to NS requests); this is the best approach to preserve privacy. But this erases

information about the relative use of the various QTYPEs, which may be interesting for researchers (for instance, if they try to follow IPv6 deployment by counting the percentage of AAAA vs. A queries). A variant of QNAME minimisation would be to keep the original QTYPE.

Another useful optimisation may be, in the spirit of the HAMMER idea [HAMMER], to probe in advance for the introduction of zone cuts where none previously existed (i.e., confirm their continued absence, or discover them).

To address the "number of queries" issue described in Section 6, a possible solution is to always use the traditional algorithm when the cache is cold and then to move to QNAME minimisation (precisely defining what is "hot" or "cold" is left to the implementer). This will decrease the privacy but will guarantee no degradation of performance.

Acknowledgments

TODO (refer to 7816)

Changes from RFC 7816

Fixed errata #4644

Moved to standards track

Authors' Addresses

Stephane Bortzmeyer
AFNIC
1, rue Stephenson
Montigny-le-Bretonneux 78180
France

Phone: +33 1 39 30 83 46
Email: bortzmeyer+ietf@nic.fr
URI: <https://www.afnic.fr/>

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

dnsop
Internet-Draft
Obsoletes: 6944 (if approved)
Intended status: Standards Track
Expires: October 22, 2019

P. Wouters
Red Hat
O. Sury
Internet Systems Consortium
April 20, 2019

Algorithm Implementation Requirements and Usage Guidance for DNSSEC
draft-ietf-dnsop-algorithm-update-10

Abstract

The DNSSEC protocol makes use of various cryptographic algorithms in order to provide authentication of DNS data and proof of non-existence. To ensure interoperability between DNS resolvers and DNS authoritative servers, it is necessary to specify a set of algorithm implementation requirements and usage guidelines to ensure that there is at least one algorithm that all implementations support. This document defines the current algorithm implementation requirements and usage guidance for DNSSEC. This document obsoletes RFC6944.

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 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 (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Updating Algorithm Implementation Requirements and Usage Guidance	2
1.2. Updating Algorithm Requirement Levels	3
1.3. Document Audience	4
2. Conventions Used in This Document	4
3. Algorithm Selection	4
3.1. DNSKEY Algorithms	4
3.2. DNSKEY Algorithm Recommendation	6
3.3. DS and CDS Algorithms	6
3.4. DS and CDS Algorithm Recommendation	7
4. Implementation Status	7
4.1. DNSKEY Algorithms	7
5. Security Considerations	8
6. Operational Considerations	8
7. IANA Considerations	9
8. Acknowledgements	9
9. References	9
9.1. Normative References	9
9.2. Informative References	10
Authors' Addresses	11

1. Introduction

The DNSSEC signing algorithms are defined by various RFCs, including [RFC4034], [RFC5155], [RFC5702], [RFC5933], [RFC6605], [RFC8080]. DNSSEC is used to provide authentication of data. To ensure interoperability, a set of "mandatory-to-implement" DNSKEY algorithms are defined. This document obsoletes [RFC6944].

1.1. Updating Algorithm Implementation Requirements and Usage Guidance

The field of cryptography evolves continuously. New, stronger algorithms appear and existing algorithms are found to be less secure than originally thought. Attacks previously thought to be computationally infeasible become more accessible as the available computational resources increase. Therefore, algorithm implementation requirements and usage guidance need to be updated from time to time to reflect the new reality. The choices for algorithms must be conservative to minimize the risk of algorithm compromise.

1.2. Updating Algorithm Requirement Levels

The mandatory-to-implement algorithm of tomorrow should already be available in most implementations of DNSSEC by the time it is made mandatory. This document attempts to identify and introduce those algorithms for future mandatory-to-implement status. There is no guarantee that algorithms in use today will become mandatory in the future. Published algorithms are continuously subjected to cryptographic attack and may become too weak, or even be completely broken, before this document is updated.

This document only provides recommendations with respect to mandatory-to-implement algorithms or algorithms so weak that these cannot be recommended. Any algorithm listed in the [DNSKEY-IANA] and [DS-IANA] registries, but not mentioned in this document, MAY be implemented. For clarification and consistency, an algorithm will be specified as MAY in this document only when it has been downgraded from a MUST or a RECOMMENDED to a MAY.

Although this document's primary purpose is to update algorithm recommendations to keep DNSSEC authentication secure over time, it also aims to do so in such a way that DNSSEC implementations remain interoperable. DNSSEC interoperability is addressed by an incremental introduction or deprecation of algorithms.

[RFC2119] considers the term SHOULD equivalent to RECOMMENDED, and SHOULD NOT equivalent to NOT RECOMMENDED. The authors of this document have chosen to use the terms RECOMMENDED and NOT RECOMMENDED, as this more clearly expresses the intent to implementers.

It is expected that deprecation of an algorithm will be performed gradually in a series of updates to this document. This provides time for various implementations to update their implemented algorithms while remaining interoperable. Unless there are strong security reasons, an algorithm is expected to be downgraded from MUST to NOT RECOMMENDED or MAY, instead of to MUST NOT. Similarly, an algorithm that has not been mentioned as mandatory-to-implement is expected to be introduced with a RECOMMENDED instead of a MUST.

Since the effect of using an unknown DNSKEY algorithm is that the zone is treated as insecure, it is recommended that algorithms downgraded to NOT RECOMMENDED or lower not be used by authoritative nameservers and DNSSEC signers to create new DNSKEYs. This will allow for deprecated algorithms to become less and less common over time. Once an algorithm has reached a sufficiently low level of deployment, it can be marked as MUST NOT, so that recursive resolvers can remove support for validating it.

Recursive nameservers are encouraged to retain support for all algorithms not marked as MUST NOT.

1.3. Document Audience

The recommendations of this document mostly target DNSSEC implementers, as implementations need to meet both high security expectations as well as high interoperability between various vendors and with different versions. Interoperability requires a smooth transition to more secure algorithms. This perspective may differ from that of a user who wishes to deploy and configure DNSSEC with only the safest algorithm. On the other hand, the comments and recommendations in this document are also expected to be useful for such users.

2. Conventions Used in This Document

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

3. Algorithm Selection

3.1. DNSKEY Algorithms

Implementation recommendations for DNSKEY algorithms [DNSKEY-IANA].

Number	Mnemonics	DNSSEC Signing	DNSSEC Validation
1	RSAMD5	MUST NOT	MUST NOT
3	DSA	MUST NOT	MUST NOT
5	RSASHA1	NOT RECOMMENDED	MUST
6	DSA-NSEC3-SHA1	MUST NOT	MUST NOT
7	RSASHA1-NSEC3-SHA1	NOT RECOMMENDED	MUST
8	RSASHA256	MUST	MUST
10	RSASHA512	NOT RECOMMENDED	MUST
12	ECC-GOST	MUST NOT	MAY
13	ECDSAP256SHA256	MUST	MUST
14	ECDSAP384SHA384	MAY	RECOMMENDED
15	ED25519	RECOMMENDED	RECOMMENDED
16	ED448	MAY	RECOMMENDED

RSAMD5 is not widely deployed and there is an industry-wide trend to deprecate MD5 usage.

RSASHA1 and RSASHA1-NSEC3-SHA1 are widely deployed, although zones deploying it are recommended to switch to ECDSAP256SHA256 as there is an industry-wide trend to move to elliptic curve cryptography. RSASHA1 does not support NSEC3. RSASHA1-NSEC3-SHA1 can be used with or without NSEC3.

DSA and DSA-NSEC3-SHA1 are not widely deployed and vulnerable to private key compromise when generating signatures using a weak or compromised random number generator.

RSASHA256 is in wide use and considered strong. It has been the default algorithm for a number of years and is now slowly being replaced with ECDSAP256SHA256 due to its shorter key and signature size, resulting in smaller DNS packets.

RSASHA512 is NOT RECOMMENDED for DNSSEC Signing because it has not seen wide deployment, but there are some deployments hence DNSSEC Validation MUST implement RSASHA512 to ensure interoperability. There is no significant difference in cryptographic strength between RSASHA512 and RSASHA256, therefore it is discouraged to use RSASHA512, as it will only make deprecation of older algorithms harder. People who wish to use a cryptographically stronger algorithm should switch to elliptic curve cryptography algorithms.

ECC-GOST (GOST R 34.10-2001) has been superseded by GOST R 34.10-2012 in [RFC7091]. GOST R 34.10-2012 hasn't been standardized for use in DNSSEC.

ECDSAP256SHA256 provides more cryptographic strength with a shorter signature length than either RSASHA256 or RSASHA512. ECDSAP256SHA256 has been widely deployed and therefore it is now at MUST level for both validation and signing. It is RECOMMENDED to use the deterministic digital signature generation procedure of the ECDSA ([RFC6979]) when implementing ECDSAP256SHA256 (and ECDSAP384SHA384).

ECDSAP384SHA384 shares the same properties as ECDSAP256SHA256, but offers a modest security advantage over ECDSAP256SHA256 (192 bits of strength versus 128 bits). For most DNSSEC applications, ECDSAP256SHA256 should be satisfactory and robust for the foreseeable future, and is therefore recommended for signing. While it is unlikely for a DNSSEC use case requiring 192-bit security strength to arise, ECDSA384SHA384 is provided for such applications and it MAY be used for signing in these cases.

ED25519 and ED448 use the Edwards-curve Digital Security Algorithm (EdDSA). There are three main advantages of EdDSA: It does not require the use of a unique random number for each signature, there are no padding or truncation issues as with ECDSA, and it is more

resilient to side-channel attacks. Furthermore, EdDSA cryptography is less prone to implementation errors ([RFC8032], [RFC8080]). It is expected that ED25519 will become the future RECOMMENDED default algorithm once there's enough support for this algorithm in the deployed DNSSEC validators.

3.2. DNSKEY Algorithm Recommendation

Due to the industry-wide trend towards elliptic curve cryptography, ECDSAP256SHA256 is the RECOMMENDED DNSKEY algorithm for use by new DNSSEC deployments, and users of RSA-based algorithms SHOULD upgrade to ECDSAP256SHA256.

3.3. DS and CDS Algorithms

Recommendations for Delegation Signer Digest Algorithms [DNSKEY-IANA] These recommendations also apply to the CDS RRTYPE as specified in [RFC7344]

Number	Mnemonics	DNSSEC Delegation	DNSSEC Validation
0	NULL (CDS only)	MUST NOT [*]	MUST NOT [*]
1	SHA-1	MUST NOT	MUST
2	SHA-256	MUST	MUST
3	GOST R 34.11-94	MUST NOT	MAY
4	SHA-384	MAY	RECOMMENDED

[*] - This is a special type of CDS record signaling removal of DS at the parent in [RFC8078].

NULL is a special case, see [RFC8078].

SHA-1 is still in wide use for DS records, so validators MUST implement validation, but it MUST NOT be used to generate new DS and CDS records. (See Operational Considerations for caveats when upgrading from SHA-1 to SHA-256 DS Algorithm.)

SHA-256 is in wide use and considered strong.

GOST R 34.11-94 has been superseded by GOST R 34.11-2012 in [RFC6986]. GOST R 34.11-2012 has not been standardized for use in DNSSEC.

SHA-384 shares the same properties as SHA-256, but offers a modest security advantage over SHA-256 (384-bits of strength versus 256-bits). For most applications of DNSSEC, SHA-256 should be

satisfactory and robust for the foreseeable future, and is therefore recommended for DS and CDS records. While it is unlikely for a DNSSEC use case requiring 384-bit security strength to arise, SHA-384 is provided for such applications and it MAY be used for generating DS and CDS records in these cases.

3.4. DS and CDS Algorithm Recommendation

Operation recommendation for new and existing deployments: SHA-256 is the RECOMMENDED DS and CDS algorithm.

4. Implementation Status

[RFC Editor Note: Please remove this entire section plus all references to [RFC7942] prior to publication as an RFC.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

4.1. DNSKEY Algorithms

The following table contains the status of support in the open-source DNS signers and validators in the current released versions as of the time writing this document. Usually, the support for specific algorithm has to be also included in the cryptographic libraries that the software use.

Mnemonics	BIND	Knot DNS	OpenDNS	PowerDNS	Unbound
RSAMD5	Y	N	Y	N	N
DSA	Y	N	Y	N	Y
RSASHA1	Y	Y	Y	Y	Y
DSA-NSEC3-SHA1	Y	N	Y	N	Y
RSASHA1-NSEC3-SHA1	Y	Y	Y	Y	Y
RSASHA256	Y	Y	Y	Y	Y
RSASHA512	Y	Y	Y	Y	Y
ECC-GOST	N	N	Y	N	Y
ECDSAP256SHA256	Y	Y	Y	Y	Y
ECDSAP384SHA384	Y	Y	Y	Y	Y
ED25519	Y	Y	N	Y	Y
ED448	N	N	N	Y	Y

5. Security Considerations

The security of cryptographic systems depends on both the strength of the cryptographic algorithms chosen and the strength of the keys used with those algorithms. The security also depends on the engineering of the protocol used by the system to ensure that there are no non-cryptographic ways to bypass the security of the overall system.

This document concerns itself with the selection of cryptographic algorithms for use in DNSSEC, specifically with the selection of "mandatory-to-implement" algorithms. The algorithms identified in this document as MUST or RECOMMENDED to implement are not known to be broken (in the cryptographic sense) at the current time, and cryptographic research so far leads us to believe that they are likely to remain secure into the foreseeable future. However, this isn't necessarily forever, and it is expected that new revisions of this document will be issued from time to time to reflect the current best practices in this area.

Retiring an algorithm too soon would result in a zone signed with the retired algorithm being downgraded to the equivalent of an unsigned zone. Therefore, algorithm deprecation must be done very slowly and only after careful consideration and measurement of its use.

6. Operational Considerations

DNSKEY algorithm rollover in a live zone is a complex process. See [RFC6781] and [RFC7583] for guidelines on how to perform algorithm rollovers.

DS algorithm rollover in a live zone is also a complex process. Upgrading algorithm at the same time as rolling the new KSK key will lead to DNSSEC validation failures, and users MUST upgrade the DS algorithm first before rolling the Key Signing Key.

7. IANA Considerations

This document makes no requests of IANA.

8. Acknowledgements

This document borrows text from RFC 4307 by Jeffrey I. Schiller of the Massachusetts Institute of Technology (MIT) and the 4307bis document by Yoav Nir, Tero Kivinen, Paul Wouters, and Daniel Migault. Much of the original text has been copied verbatim.

We wish to thank Michael Sinatra, Roland van Rijswijk-Deij, Olafur Gudmundsson, Paul Hoffman, Evan Hunt, and Peter Yee for their feedback.

Kudos to Roy Arends for bringing the DS rollover issue to light.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <<https://www.rfc-editor.org/info/rfc5155>>.
- [RFC5702] Jansen, J., "Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC", RFC 5702, DOI 10.17487/RFC5702, October 2009, <<https://www.rfc-editor.org/info/rfc5702>>.

- [RFC6605] Hoffman, P. and W. Wijngaards, "Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC", RFC 6605, DOI 10.17487/RFC6605, April 2012, <<https://www.rfc-editor.org/info/rfc6605>>.
- [RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <<https://www.rfc-editor.org/info/rfc6979>>.
- [RFC6986] Dolmatov, V., Ed. and A. Degtyarev, "GOST R 34.11-2012: Hash Function", RFC 6986, DOI 10.17487/RFC6986, August 2013, <<https://www.rfc-editor.org/info/rfc6986>>.
- [RFC7344] Kumari, W., Gudmundsson, O., and G. Barwood, "Automating DNSSEC Delegation Trust Maintenance", RFC 7344, DOI 10.17487/RFC7344, September 2014, <<https://www.rfc-editor.org/info/rfc7344>>.
- [RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <<https://www.rfc-editor.org/info/rfc8032>>.
- [RFC8078] Gudmundsson, O. and P. Wouters, "Managing DS Records from the Parent via CDS/CDNSKEY", RFC 8078, DOI 10.17487/RFC8078, March 2017, <<https://www.rfc-editor.org/info/rfc8078>>.
- [RFC8080] Sury, O. and R. Edmonds, "Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC", RFC 8080, DOI 10.17487/RFC8080, February 2017, <<https://www.rfc-editor.org/info/rfc8080>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

- [RFC5933] Dolmatov, V., Ed., Chuprina, A., and I. Ustinov, "Use of GOST Signature Algorithms in DNSKEY and RRSIG Resource Records for DNSSEC", RFC 5933, DOI 10.17487/RFC5933, July 2010, <<https://www.rfc-editor.org/info/rfc5933>>.

- [RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <<https://www.rfc-editor.org/info/rfc6781>>.
- [RFC6944] Rose, S., "Applicability Statement: DNS Security (DNSSEC) DNSKEY Algorithm Implementation Status", RFC 6944, DOI 10.17487/RFC6944, April 2013, <<https://www.rfc-editor.org/info/rfc6944>>.
- [RFC7091] Dolmatov, V., Ed. and A. Degtyarev, "GOST R 34.10-2012: Digital Signature Algorithm", RFC 7091, DOI 10.17487/RFC7091, December 2013, <<https://www.rfc-editor.org/info/rfc7091>>.
- [RFC7583] Morris, S., Ihren, J., Dickinson, J., and W. Mekking, "DNSSEC Key Rollover Timing Considerations", RFC 7583, DOI 10.17487/RFC7583, October 2015, <<https://www.rfc-editor.org/info/rfc7583>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [DNSKEY-IANA] "DNSKEY Algorithms", <<http://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml>>.
- [DS-IANA] "Delegation Signer Digest Algorithms", <<http://www.iana.org/assignments/ds-rr-types/ds-rr-types.xhtml>>.

Authors' Addresses

Paul Wouters
Red Hat
CA

EMail: pwouters@redhat.com

Ondrej Sury
Internet Systems Consortium
CZ

EMail: ondrej@isc.org

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

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

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2020.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction
 - 1.1. Overview
 - 1.2. Terminology
2. The ANAME resource record
 - 2.1. Presentation and wire format

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

1. Introduction

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

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

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

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

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

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

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

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

1.1. Overview

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

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

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

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

1.2. Terminology

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

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

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

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

Section 3).

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

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

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

2. The ANAME resource record

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

2.1. Presentation and wire format

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

```
owner ttl class ANAME target
```

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

2.2. Coexistence with other types

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

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

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

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

3. Substituting ANAME sibling address records

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

The following steps MUST be performed for each address type:

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

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

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

4. ANAME processing by primary masters

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

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

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

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

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

4.1. Zone transfers

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

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

Secondary servers rely on zone transfers to obtain sibling address

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

4.2. DNSSEC

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

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

4.3. TTLs

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

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

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

5. ANAME processing by resolvers

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

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

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

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

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

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

6. Query processing

6.1. Authoritative servers

6.1.1. Address queries

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

6.1.2. ANAME queries

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

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

6.2. Resolvers

6.2.1. Address queries

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

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

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

6.2.2. ANAME queries

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

7. IANA considerations

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

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

8. Security considerations

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

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

9. Acknowledgments

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

10. Changes since the last revision

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

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

10.1. Version -04

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

10.2. Version -03

- o Grammar improvements (Olli Vanhoja)

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

10.3. Version -02

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

11. References

11.1. Normative References

- [RFC1033] Lottor, M., "Domain Administrators Operations Guide", RFC 1033, DOI 10.17487/RFC1033, November 1987, <<https://www.rfc-editor.org/info/rfc1033>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<https://www.rfc-editor.org/info/rfc2136>>.
- [RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <<https://www.rfc-editor.org/info/rfc3597>>.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.
- [RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <<https://www.rfc-editor.org/info/rfc7871>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.

11.2. Informative References

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

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

- [RFC0973] Mockapetris, P., "Domain system changes and observations", RFC 973, DOI 10.17487/RFC0973, January 1986, <<https://www.rfc-editor.org/info/rfc973>>.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <<https://www.rfc-editor.org/info/rfc1996>>.
- [RFC2065] Eastlake 3rd, D. and C. Kaufman, "Domain Name System Security Extensions", RFC 2065, DOI 10.17487/RFC2065, January 1997, <<https://www.rfc-editor.org/info/rfc2065>>.
- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <<https://www.rfc-editor.org/info/rfc2308>>.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <<https://www.rfc-editor.org/info/rfc3493>>.

11.3. URIs

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

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

Appendix A. Implementation status

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

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

Appendix B. Historical note

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

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

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

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

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

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

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

Appendix C. On preserving TTLs

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

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

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

C.1. Query bunching

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

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

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

C.2. Upstream caches

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

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

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

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

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

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

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

C.3. ANAME chains

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

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

C.4. ANAME substitution inside the name server

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

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

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

C.5. TTLs and zone transfers

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

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

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

Appendix D. Alternative setups

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

D.1. Reducing query volume

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

D.2. Zone transfer scalability

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

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

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

D.3. Tailored responses

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

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

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

Appendix E. ANAME loops

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

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

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

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

Authors' Addresses

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

Email: dot@dotat.at

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

Email: each@isc.org

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

Email: peter.van.dijk@powerdns.com

Anthony Eden
DNSimple
Boston, MA USA

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

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

Email: matthijs@isc.org

Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: January 3, 2019

L. Song
Beijing Internet Institute
P. Vixie
TISF
S. Kerr
July 2, 2018

An Proxy Use Case of DNS over HTTPS
draft-ietf-dnsop-dns-wireformat-http-03

Abstract

This memo introduces a DNS proxy use case to tunnel DNS query and response using DNS over HTTPs (DOH) protocol, a newly proposed DNS transport. The proxy use case is useful as a incremental adoption tool when DOH is not widely available in old-transport client and server.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Use case description	3
3. Original transport indicator in DOH proxy	4
4. Implementation considerations	4
5. Security Considerations	4
6. IANA considerations	5
7. Acknowledgments	5
8. References	5
Authors' Addresses	5

1. Introduction

RFC 1035 [RFC1035] specifies the wire format for DNS messages. It also specifies DNS transport on UDP and TCP on port 53, which is still used today. To enhance the availability of honest DNS, a new DNS transport: DNS over HTTPS (DOH) [I-D.ietf-doh-dns-over-https] is proposed which transport DNS over HTTPS, in a way to cure DNS's long-time suffering from on-path attack by spoofing and blocking.

This memo introduces a DNS proxy use case to leverage the DOH protocol as a substrate to tunnel DNS data over HTTPS which is called DOH proxy in the rest of the document. It is useful especially when most DNS stub-resolvers and far-end servers are not aware the new DOH protocol, but a public or private proxy using DOH can be deployed and offer DOH capacity to users to bypass the networks where DNS is not working properly.

Just as a normal DNS proxy described in [RFC5625], DOH proxy works as a simple DNS forwarder keeping the transparency principle, so any "hop-by-hop" mechanisms or newly introduced protocol extensions operate as if the proxy were not there.

In order to keep the transparency of DOH proxy, a new variable "proto" in URI Template is defined for DOH proxy use case. It allows the proxy server use the same transport protocol (UDP or TCP) to forward DNS query to far-end server just as the stub-client does without DOH proxy.

May REMOVE BEFORE PUBLICATION: Comparing using a general VPN, the DOH proxy can work on an actual HTTP server, so it can be hosted on a machine that also serves web pages. This means that DNS over HTTP is slightly more "stealthy" than a VPN, in that it can be indistinguishable from normal web traffic.

2. Use case description

The typical scenario is that a DOH proxy sitting between stub-resolver and the recursive server. The stub-resolver is configured sending DNS query to a DOH proxy and expects reply from the same DOH proxy. Just as a normal DNS proxy described in [RFC5625], DOH proxy works as a simple DNS forwarder keeping the transparency principle. The only difference is DOH proxy consist two part, a proxy client as a initiator of DOH tunnel and a proxy server as a terminator. The proxy client speaks DOH with proxy server carrying the same DNS query received from stub-resolver. The proxy server will forward the exact DNS query received from stub-resolver to the configued recursive server.

To keep the transparency principle of DOH proxy, any "hop-by-hop" mechanisms or newly introduced protocol extensions operate as if the DOH proxy were not there. Different from the native DOH protocol, in DOH proxy use case, there should be a indication introduced for proxy client to tell the proxy server original transport (UDP or TCP) the stub-resolver uses to send DNS query to proxy client.

For example if the proxy client receives the query via UDP, then it will notify the proxy server with a "proto=udp" indicator which is defined in Section 3. If proxy client receives the query via TCP, then it will carry a "proto=tcp" indicator with the same DNS query without the two-byte length field defined in DNS over TCP [section 4.2.2 in [RFC1035]].

Besides the original transport indicator, as specified in DOH document, the proxy server MUST be able to process both "application/dns-message" request messages and forward the query to a configured recursive server using the same transport between sub-resolver and proxy client. The response will be delivered back to sub-resolver accordingly. In DOH proxy use case, each DNS query-response pair is mapped into a DOH query-response pair. And the transport for DNS query and response MUST be the same.

It is possible that a proxy client as a module can be deployed in the same host with the sub-client listening to a loop-back address. A proxy server can be implemented that way to host a recursive DNS process as well. The can be combined to form four deployment scenarios of DOH proxy use case.

It is also possible to use the proxy server as a regular web server at the same time that is acting as a proxy server.

Note that the proxy client will face the same bootstrapping problem described in DOH when the HTTPs request needs to resolve the name of

server and send the request to on IP address. The strategy is either use the IP directly or use another resolver (like the normal DHCP-supplied resolver) to lookup the IP of the server.

3. Original transport indicator in DOH proxy

In DOH document[I-D.ietf-doh-dns-over-https], the HTTP request uses a URI defined by the DOH server through the use of a URI Template in which no variables is defined. In this document, a new variable "proto" is defined as the indicator of original transport. For example, The URI "https://example.com/proxy_dns?proto=tcp" will cause the server to make a request using TCP. And the URL "https://example.com/proxy_dns?proto=udp" will cause the server to make a request using UDP.

4. Implementation considerations

The DOH proxy may return TC bit to the sub-resolver which will cause TCP fallback starting from the sub-resolver. An alternative advised is that the proxy has to have sufficient smarts to recognize the returned TC bit and re-issue the request over TCP to the back-end DNS server.

Another implementation is suggested that DOH proxy server has a pool of TCP connections from the proxy to the back-end DNS server(s), over which incoming requests can be multiplexed.

5. Security Considerations

The DOH proxy use case does not introduce new protocol and any new security considerations since it is built on the DNS over HTTPS protocols. All security considerations and recommendations apply in DOH proxy use case.

Since DOH proxy is a also a special DNS proxy, the security recommendations of DNS proxy RFC 5625 [RFC5625] also apply in DOH proxy use case.

Note that the ability to perform DNS queries in this way may allow users to bypass local DNS policy. This may be problematic in any environment where administrators need to enforce specific DNS behavior, such as an enterprise environment. The protocol outlined here does not introduce any new capabilities in this area, but by creating a more standardized way of doing this it may cause operational problems for enterprise administrators.

6. IANA considerations

No IANA considerations for DOH proxy

7. Acknowledgments

Thanks to Bob Harold, Paul Hoffman, Julian Reschke, Martin Thomson, Tony Finch, Ray Bellis and Erik Kline for their review and comments.

8. References

- [I-D.ietf-doh-dns-over-https]
Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", draft-ietf-doh-dns-over-https-12 (work in progress), June 2018.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <<https://www.rfc-editor.org/info/rfc5625>>.

Authors' Addresses

Linjian Song
Beijing Internet Institute
2nd Floor, Building 5, No.58 Jing Hai Wu Lu, BDA
Beijing 100176
P. R. China

Email: songlinjian@gmail.com
URI: <http://www.biigroup.com/>

Paul Vixie
TISF
11400 La Honda Road
Woodside, California 94062
US

Email: vixie@tisf.net
URI: <http://www.redbarn.org/>

Shane Kerr
Antoon Coolenlaan 41
Uithoorn 1422 GN
NL

Email: shane@time-travellers.org

Internet Engineering Task Force
Internet-Draft
Obsoletes: 2845, 4635 (if approved)
Intended status: Standards Track
Expires: January 11, 2021

F. Dupont
S. Morris
ISC
P. Vixie
Farsight
D. Eastlake 3rd
Futurewei
O. Gudmundsson
Cloudflare
B. Wellington
Akamai
July 10, 2020

Secret Key Transaction Authentication for DNS (TSIG)
draft-ietf-dnsop-rfc2845bis-09

Abstract

This document describes a protocol for transaction level authentication using shared secrets and one way hashing. It can be used to authenticate dynamic updates to a DNS zone as coming from an approved client, or to authenticate responses as coming from an approved name server.

No recommendation is made here for distributing the shared secrets: it is expected that a network administrator will statically configure name servers and clients using some out of band mechanism.

This document obsoletes RFC2845 and RFC4635.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 11, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Background	3
1.2. Protocol Overview	4
1.3. Document History	4
2. Key Words	5
3. Assigned Numbers	5
4. TSIG RR Format	5
4.1. TSIG RR Type	5
4.2. TSIG Record Format	6
4.3. MAC Computation	8
4.3.1. Request MAC	8
4.3.2. DNS Message	9
4.3.3. TSIG Variables	9
5. Protocol Details	10
5.1. Generation of TSIG on Requests	10
5.2. Server Processing of Request	10
5.2.1. Key Check and Error Handling	11
5.2.2. MAC Check and Error Handling	11
5.2.3. Time Check and Error Handling	12

5.2.4. Truncation Check and Error Handling	13
5.3. Generation of TSIG on Answers	13
5.3.1. TSIG on TCP Connections	13
5.3.2. Generation of TSIG on Error Returns	14
5.4. Client Processing of Answer	15
5.4.1. Key Error Handling	15
5.4.2. MAC Error Handling	15
5.4.3. Time Error Handling	15
5.4.4. Truncation Error Handling	16
5.5. Special Considerations for Forwarding Servers	16
6. Algorithms and Identifiers	16
7. TSIG Truncation Policy	17
8. Shared Secrets	18
9. IANA Considerations	18
10. Security Considerations	19
10.1. Issue Fixed in this Document	20
10.2. Why not DNSSEC?	20
11. References	21
11.1. Normative References	21
11.2. Informative References	22
Appendix A. Acknowledgments	24
Appendix B. Change History (to be removed before publication) .	24
Authors' Addresses	28

1. Introduction

1.1. Background

The Domain Name System (DNS, [RFC1034], [RFC1035]) is a replicated hierarchical distributed database system that provides information fundamental to Internet operations, such as name to address translation and mail handling information.

This document specifies use of a message authentication code (MAC), generated using certain keyed hash functions, to provide an efficient means of point-to-point authentication and integrity checking for DNS transactions. Such transactions include DNS update requests and responses for which this can provide a lightweight alternative to the secure DNS dynamic update protocol described by [RFC3007].

A further use of this mechanism is to protect zone transfers. In this case the data covered would be the whole zone transfer including any glue records sent. The protocol described by DNSSEC ([RFC4033], [RFC4034], [RFC4035]) does not protect glue records and unsigned records.

The authentication mechanism proposed here provides a simple and efficient authentication between clients and servers, by using shared

secret keys to establish a trust relationship between two entities. Such keys must be protected in a manner similar to private keys, lest a third party masquerade as one of the intended parties (by forging the MAC). The proposal is unsuitable for general server to server authentication and for servers which speak with many other servers, since key management would become unwieldy with the number of shared keys going up quadratically. But it is suitable for many resolvers on hosts that only talk to a few recursive servers.

1.2. Protocol Overview

Secret Key Transaction Authentication makes use of signatures on messages sent between the parties involved (e.g. resolver and server). These are known as "transaction signatures", or TSIG. For historical reasons, in this document they are referred to as message authentication codes (MAC).

Use of TSIG presumes prior agreement between the two parties involved (e.g., resolver and server) as to any algorithm and key to be used. The way that this agreement is reached is outside the scope of the document.

A DNS message exchange involves the sending of a query and the receipt of one of more DNS messages in response. For the query, the MAC is calculated based on the hash of the contents and the agreed TSIG key. The MAC for the response is similar, but also includes the MAC of the query as part of the calculation. Where a response comprises multiple packets, the calculation of the MAC associated with the second and subsequent packets includes in its inputs the MAC for the preceding packet. In this way it is possible to detect any interruption in the packet sequence, although not its premature termination.

The MAC is contained in a TSIG resource record included in the Additional Section of the DNS message.

1.3. Document History

TSIG was originally specified by [RFC2845]. In 2017, two nameserver implementations strictly following that document (and the related [RFC4635]) were discovered to have security problems related to this feature ([CVE-2017-3142], [CVE-2017-3143], [CVE-2017-11104]). The implementations were fixed but, to avoid similar problems in the future, the two documents were updated and merged, producing this revised specification for TSIG.

While TSIG implemented according to this RFC provides for enhanced security, there are no changes in interoperability. TSIG is on the

wire still the same mechanism described in [RFC2845]; only the checking semantics have been changed. See Section 10.1 for further details.

2. Key Words

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

3. Assigned Numbers

This document defines the following RR type and associated value:

TSIG (250)

In addition, the document also defines the following DNS RCODEs and associated names:

16 (BADSIG)
17 (BADKEY)
18 (BADTIME)
22 (BADTRUNC)

(See [RFC6895] Section 2.3 concerning the assignment of the value 16 to BADSIG.)

These RCODES may appear within the "Error" field of a TSIG RR.

4. TSIG RR Format

4.1. TSIG RR Type

To provide secret key authentication, we use an RR type whose mnemonic is TSIG and whose type code is 250. TSIG is a meta-RR and MUST NOT be cached. TSIG RRs are used for authentication between DNS entities that have established a shared secret key. TSIG RRs are dynamically computed to cover a particular DNS transaction and are not DNS RRs in the usual sense.

As the TSIG RRs are related to one DNS request/response, there is no value in storing or retransmitting them, thus the TSIG RR is discarded once it has been used to authenticate a DNS message.

4.2. TSIG Record Format

The fields of the TSIG RR are described below. As is usual, all multi-octet integers in the record are sent in network byte order (see [RFC1035] 2.3.2).

NAME The name of the key used, in domain name syntax. The name should reflect the names of the hosts and uniquely identify the key among a set of keys these two hosts may share at any given time. For example, if hosts A.site.example and B.example.net share a key, possibilities for the key name include <id>.A.site.example, <id>.B.example.net, and <id>.A.site.example.B.example.net. It should be possible for more than one key to be in simultaneous use among a set of interacting hosts. This allows for periodic key rotation as per best operational practices, as well as algorithm agility as indicated by [BCP201].

The name may be used as a local index to the key involved but it is recommended that it be globally unique. Where a key is just shared between two hosts, its name actually need only be meaningful to them but it is recommended that the key name be mnemonic and incorporates the names of participating agents or resources as suggested above.

TYPE This MUST be TSIG (250: Transaction SIGNature)

CLASS This MUST be ANY

TTL This MUST be 0

RdLen (variable)

RDATA The RDATA for a TSIG RR consists of a number of fields, described below:

request, it is set equal to the DNS message ID. In a TSIG attached to a response - or in cases such as the forwarding of a dynamic update request - the field contains the ID of the original DNS request.

- * Error - in responses, an unsigned 16-bit integer containing the extended RCODE covering TSIG processing. In requests, this MUST be zero.
- * Other Len - an unsigned 16-bit integer specifying the length of the "Other Data" field in octets.
- * Other Data - additional data relevant to the TSIG record. In responses, this will be empty (i.e. "Other Len" will be zero) unless the content of the Error field is BADTIME, in which case it will be a 48-bit unsigned integer containing the server's current time as the number of seconds since 00:00 on 1970-01-01 UTC, ignoring leap seconds (see Section 5.2.3). This document assigns no meaning to its contents in requests.

4.3. MAC Computation

When generating or verifying the contents of a TSIG record, the data listed in the rest of this section are passed, in the order listed below, as input to MAC computation. The data are passed in network byte order or wire format, as appropriate, and are fed into the hashing function as a continuous octet sequence with no inter-field separator or padding.

4.3.1. Request MAC

Only included in the computation of a MAC for a response message (or the first message in a multi-message response), the validated request MAC MUST be included in the MAC computation. If the request MAC failed to validate, an unsigned error message MUST be returned instead. (Section 5.3.2).

The request's MAC, comprising the following fields, is digested in wire format:

Field	Type	Description
MAC Length	Unsigned 16-bit integer	in network byte order
MAC Data	octet sequence	exactly as transmitted

Special considerations apply to the TSIG calculation for the second and subsequent messages a response that consists of multiple DNS

messages (e.g. a zone transfer). These are described in Section 5.3.1.

4.3.2. DNS Message

The DNS message used in the MAC computation is a whole and complete DNS message in wire format.

When creating a TSIG, it is the message before the TSIG RR has been added to the additional data section and before the DNS Message Header's ARCOUNT field has been incremented to contain the TSIG RR.

When verifying an incoming message, it is the message after the TSIG RR has been removed and the ARCOUNT field decremented. If the message ID differs from the original message ID, the original message ID is substituted for the message ID. (This could happen, for example, when forwarding a dynamic update request.)

4.3.3. TSIG Variables

Also included in the digest is certain information present in the TSIG RR. Adding this data provides further protection against an attempt to interfere with the message.

Source	Field Name	Notes
TSIG RR	NAME	Key name, in canonical wire format
TSIG RR	CLASS	(Always ANY in the current specification)
TSIG RR	TTL	(Always 0 in the current specification)
TSIG RDATA	Algorithm Name	in canonical wire format
TSIG RDATA	Time Signed	in network byte order
TSIG RDATA	Fudge	in network byte order
TSIG RDATA	Error	in network byte order
TSIG RDATA	Other Len	in network byte order
TSIG RDATA	Other Data	exactly as transmitted

Table 1

The RR RDLEN and RDATA MAC Length are not included in the input to MAC computation since they are not guaranteed to be knowable before the MAC is generated.

The Original ID field is not included in this section, as it has already been substituted for the message ID in the DNS header and hashed.

For each label type, there must be a defined "Canonical wire format" that specifies how to express a label in an unambiguous way. For

label type 00, this is defined in [RFC4034] Section 6.2. The use of label types other than 00 is not defined for this specification.

4.3.3.1. Time Values Used in TSIG Calculations

The data digested includes the two timer values in the TSIG header in order to defend against replay attacks. If this were not done, an attacker could replay old messages but update the "Time Signed" and "Fudge" fields to make the message look new. This data is named "TSIG Timers", and for the purpose of MAC calculation, they are hashed in their "on the wire" format, in the following order: first Time Signed, then Fudge.

5. Protocol Details

5.1. Generation of TSIG on Requests

Once the outgoing record has been constructed, the client performs the keyed hash (HMAC) computation, appends a TSIG record with the calculated MAC to the Additional Data section (incrementing the ARCOUNT to reflect the additional RR), and transmits the request to the server. This TSIG record MUST be the only TSIG RR in the message and MUST be last record in the Additional Data section. The client MUST store the MAC and the key name from the request while awaiting an answer.

The digest components for a request are:

- DNS Message (request)
- TSIG Variables (request)

5.2. Server Processing of Request

If an incoming message contains a TSIG record, it MUST be the last record in the additional section. Multiple TSIG records are not allowed. If multiple TSIG records are detected or a TSIG record is present in any other position, the DNS message is dropped and a response with RCODE 1 (FORMERR) MUST be returned. Upon receipt of a message with exactly one correctly placed TSIG RR, a copy of the TSIG RR is stored, and the TSIG RR is removed from the DNS Message, and decremented out of the DNS message header's ARCOUNT.

If the TSIG RR cannot be interpreted, the server MUST regard the message as corrupt and return a FORMERR to the server. Otherwise the server is REQUIRED to return a TSIG RR in the response.

To validate the received TSIG RR, the server MUST perform the following checks in the following order:

1. Check KEY
2. Check MAC
3. Check TIME values
4. Check Truncation policy

5.2.1. Key Check and Error Handling

If a non-forwarding server does not recognize the key or algorithm used by the client (or recognizes the algorithm but does not implement it), the server MUST generate an error response with RCODE 9 (NOTAUTH) and TSIG ERROR 17 (BADKEY). This response MUST be unsigned as specified in Section 5.3.2. The server SHOULD log the error. (Special considerations apply to forwarding servers, see Section 5.5.)

5.2.2. MAC Check and Error Handling

Using the information in the TSIG, the server MUST verify the MAC by doing its own calculation and comparing the result with the MAC received. If the MAC fails to verify, the server MUST generate an error response as specified in Section 5.3.2 with RCODE 9 (NOTAUTH) and TSIG ERROR 16 (BADSIG). This response MUST be unsigned as specified in Section 5.3.2. The server SHOULD log the error.

5.2.2.1. MAC Truncation

When space is at a premium and the strength of the full length of a MAC is not needed, it is reasonable to truncate the keyed hash and use the truncated value for authentication. HMAC SHA-1 truncated to 96 bits is an option available in several IETF protocols, including IPsec and TLS. However, while this option is kept for backwards compatibility, it may not provide a security level appropriate for all cases in the modern environment. In these cases, it is preferable to use a hashing algorithm such as SHA-256-128, SHA-384-192 or SHA-512-256 [RFC4868].

Processing of a truncated MAC follows these rules:

1. If "MAC size" field is greater than keyed hash output length:

This case MUST NOT be generated and, if received, MUST cause the DNS message to be dropped and RCODE 1 (FORMERR) to be returned.
2. If "MAC size" field equals keyed hash output length:

The entire output keyed hash output is present and used.

3. "MAC size" field is less than the larger of 10 (octets) and half the length of the hash function in use:

With the exception of certain TSIG error messages described in Section 5.3.2, where it is permitted that the MAC size be zero, this case MUST NOT be generated and, if received, MUST cause the DNS message to be dropped and RCODE 1 (FORMERR) to be returned.

4. Otherwise:

This is sent when the signer has truncated the keyed hash output to an allowable length, as described in [RFC2104], taking initial octets and discarding trailing octets. TSIG truncation can only be to an integral number of octets. On receipt of a DNS message with truncation thus indicated, the locally calculated MAC is similarly truncated and only the truncated values are compared for authentication. The request MAC used when calculating the TSIG MAC for a reply is the truncated request MAC.

5.2.3. Time Check and Error Handling

If the server time is outside the time interval specified by the request (which is: Time Signed, plus/minus Fudge), the server MUST generate an error response with RCODE 9 (NOTAUTH) and TSIG ERROR 18 (BADTIME). The server SHOULD also cache the most recent Time Signed value in a message generated by a key, and SHOULD return BADTIME if a message received later has an earlier Time Signed value. A response indicating a BADTIME error MUST be signed by the same key as the request. It MUST include the client's current time in the Time Signed field, the server's current time (an unsigned 48-bit integer) in the Other Data field, and 6 in the Other Len field. This is done so that the client can verify a message with a BADTIME error without the verification failing due to another BADTIME error. In addition, the Fudge field MUST be set to the fudge value received from the client. The data signed is specified in Section 5.3.2. The server SHOULD log the error.

Caching the most recent Time Signed value and rejecting requests with an earlier one could lead to valid messages being rejected if transit through the network led to UDP packets arriving in a different order to the one in which they were sent. Implementations should be aware of this possibility and be prepared to deal with it, e.g. by retransmitting the rejected request with a new TSIG once outstanding requests have completed or the time given by their Time Signed plus fudge value has passed. If implementations do retry requests in these cases, a limit SHOULD be placed on the maximum number of retries.

5.2.4. Truncation Check and Error Handling

If a TSIG is received with truncation that is permitted under Section 5.2.2.1 above but the MAC is too short for the local policy in force, an RCODE 9 (NOTAUTH) and TSIG ERROR 22 (BADTRUNC) MUST be returned. The server SHOULD log the error.

5.3. Generation of TSIG on Answers

When a server has generated a response to a signed request, it signs the response using the same algorithm and key. The server MUST NOT generate a signed response to a request if either the KEY is invalid (e.g. key name or algorithm name are unknown), or the MAC fails validation: see Section 5.3.2 for details of responding in these cases.

It also MUST NOT not generate a signed response to an unsigned request, except in the case of a response to a client's unsigned TKEY request if the secret key is established on the server side after the server processed the client's request. Signing responses to unsigned TKEY requests MUST be explicitly specified in the description of an individual secret key establishment algorithm [RFC3645].

The digest components used to generate a TSIG on a response are:

- Request MAC
- DNS Message (response)
- TSIG Variables (response)

(This calculation is different for the second and subsequent message in a multi-message answer, see below.)

If addition of the TSIG record will cause the message to be truncated, the server MUST alter the response so that a TSIG can be included. This response consists of only the question and a TSIG record, and has the TC bit set and an RCODE of 0 (NOERROR). The client SHOULD at this point retry the request using TCP (as per [RFC1035] 4.2.2).

5.3.1. TSIG on TCP Connections

A DNS TCP session such as a zone transfer can include multiple DNS messages. Using TSIG on such a connection can protect the connection from attack and provide data integrity. The TSIG MUST be included on all DNS messages in the response. For backward compatibility, a client which receives DNS messages and verifies TSIG MUST accept up to 99 intermediary messages without a TSIG and MUST verify that both the first and last message contain a TSIG.

The first message is processed as a standard answer (see Section 5.3) but subsequent messages have the following digest components:

- Prior MAC (running)
- DNS Messages (any unsigned messages since the last TSIG)
- TSIG Timers (current message)

The "Prior MAC" is the MAC from the TSIG attached to the last message containing a TSIG. "DNS Messages" comprises the concatenation (in message order) of all messages after the last message that included a TSIG and includes the current message. "TSIG timers" comprises the "Time Signed" and "Fudge" fields (in that order) pertaining to the message for which the TSIG is being created: this means that the successive TSIG records in the stream will have non-decreasing "Time Signed" fields. Note that only the timers are included in the second and subsequent messages, not all the TSIG variables.

This allows the client to rapidly detect when the session has been altered; at which point it can close the connection and retry. If a client TSIG verification fails, the client **MUST** close the connection. If the client does not receive TSIG records frequently enough (as specified above) it **SHOULD** assume the connection has been hijacked and it **SHOULD** close the connection. The client **SHOULD** treat this the same way as they would any other interrupted transfer (although the exact behavior is not specified).

5.3.2. Generation of TSIG on Error Returns

When a server detects an error relating to the key or MAC in the incoming request, the server **SHOULD** send back an unsigned error message (MAC size == 0 and empty MAC). It **MUST NOT** send back a signed error message.

If an error is detected relating to the TSIG validity period or the MAC is too short for the local policy, the server **SHOULD** send back a signed error message. The digest components are:

- Request MAC (if the request MAC validated)
- DNS Message (response)
- TSIG Variables (response)

The reason that the request MAC is not included in this MAC in some cases is to make it possible for the client to verify the error. If the error is not a TSIG error the response **MUST** be generated as specified in Section 5.3.

5.4. Client Processing of Answer

When a client receives a response from a server and expects to see a TSIG, it first checks if the TSIG RR is present in the response. If not, the response is treated as having a format error and is discarded.

If the TSIG RR is present, the client performs the same checks as described in Section 5.2. If the TSIG RR is unsigned as specified in Section 5.3.2 or does not validate, the message **MUST** be discarded unless the RCODE is 9 (NOTAUTH). In this case, the client **SHOULD** attempt to verify the response as if it were a TSIG error, as described in the following subsections.

Regardless of the RCODE, a message containing a TSIG RR that is unsigned as specified in Section 5.3.2 or which fails verification **SHOULD NOT** be considered an acceptable response as it may have been spoofed or manipulated. Instead, the client **SHOULD** log an error and continue to wait for a signed response until the request times out.

5.4.1. Key Error Handling

If an RCODE on a response is 9 (NOTAUTH), but the response TSIG validates and the TSIG key is recognized by the client but different from that used on the request, then this is a Key Error. The client **MAY** retry the request using the key specified by the server. However, this should never occur, as a server **MUST NOT** sign a response with a different key to that used to sign the request.

5.4.2. MAC Error Handling

If the response RCODE is 9 (NOTAUTH) and TSIG ERROR is 16 (BADSIG), this is a MAC error, and client **MAY** retry the request with a new request ID but it would be better to try a different shared key if one is available. Clients **SHOULD** keep track of how many MAC errors are associated with each key. Clients **SHOULD** log this event.

5.4.3. Time Error Handling

If the response RCODE is 9 (NOTAUTH) and the TSIG ERROR is 18 (BADTIME), or the current time does not fall in the range specified in the TSIG record, then this is a Time error. This is an indication that the client and server clocks are not synchronized. In this case the client **SHOULD** log the event. DNS resolvers **MUST NOT** adjust any clocks in the client based on BADTIME errors, but the server's time in the Other Data field **SHOULD** be logged.

5.4.4. Truncation Error Handling

If the response RCODE is 9 (NOTAUTH) and the TSIG ERROR is 22 (BADTRUNC) then this is a Truncation error. The client MAY retry with a lesser truncation up to the full HMAC output (no truncation), using the truncation used in the response as a hint for what the server policy allowed (Section 7). Clients SHOULD log this event.

5.5. Special Considerations for Forwarding Servers

A server acting as a forwarding server of a DNS message SHOULD check for the existence of a TSIG record. If the name on the TSIG is not of a secret that the server shares with the originator the server MUST forward the message unchanged including the TSIG. If the name of the TSIG is of a key this server shares with the originator, it MUST process the TSIG. If the TSIG passes all checks, the forwarding server MUST, if possible, include a TSIG of its own, to the destination or the next forwarder. If no transaction security is available to the destination and the message is a query then, if the corresponding response has the AD flag (see [RFC4035]) set, the forwarder MUST clear the AD flag before adding the TSIG to the response and returning the result to the system from which it received the query.

6. Algorithms and Identifiers

The only message digest algorithm specified in the first version of these specifications [RFC2845] was "HMAC-MD5" (see [RFC1321], [RFC2104]). Although a review of its security some years ago [RFC6151] concluded that "it may not be urgent to remove HMAC-MD5 from the existing protocols", with the availability of more secure alternatives the opportunity has been taken to make the implementation of this algorithm optional.

[RFC4635] added mandatory support in TSIG for SHA-1 [FIPS180-4], [RFC3174]. SHA-1 collisions have been demonstrated [SHA1SHAMBLES] so the MD5 security considerations described in section 2 of [RFC6151] apply to SHA-1 in a similar manner. Although support for hmac-sha1 in TSIG is still mandatory for compatibility reasons, existing uses SHOULD be replaced with hmac-sha256 or other SHA-2 digest algorithms [FIPS180-4], [RFC3874], [RFC6234].

Use of TSIG between two DNS agents is by mutual agreement. That agreement can include the support of additional algorithms and criteria as to which algorithms and truncations are acceptable, subject to the restriction and guidelines in Section 5.2.2.1 above. Key agreement can be by the TKEY mechanism [RFC2930] or some other mutually agreeable method.

Implementations that support TSIG MUST also implement HMAC SHA1 and HMAC SHA256 and MAY implement gss-tsig and the other algorithms listed below. SHA-1 truncated to 96 bits (12 octets) SHOULD be implemented.

Name	Implementation	Use
HMAC-MD5.SIG-ALG.REG.INT	MAY	MUST NOT
gss-tsig	MAY	MAY
hmac-sha1	MUST	NOT RECOMMENDED
hmac-sha224	MAY	MAY
hmac-sha256	MUST	RECOMMENDED
hmac-sha256-128	MAY	MAY
hmac-sha384	MAY	MAY
hmac-sha384-192	MAY	MAY
hmac-sha512	MAY	MAY
hmac-sha512-256	MAY	MAY

Table 2

7. TSIG Truncation Policy

As noted above, two DNS agents (e.g., resolver and server) must mutually agree to use TSIG. Implicit in such an "agreement" are criteria as to acceptable keys and algorithms and, with the extensions in this document, truncations. Local policies MAY require the rejection of TSIGs, even though they use an algorithm for which implementation is mandatory.

When a local policy permits acceptance of a TSIG with a particular algorithm and a particular non-zero amount of truncation, it SHOULD also permit the use of that algorithm with lesser truncation (a longer MAC) up to the full keyed hash output.

Regardless of a lower acceptable truncated MAC length specified by local policy, a reply SHOULD be sent with a MAC at least as long as that in the corresponding request. Note if the request specified a MAC length longer than the keyed hash output it will be rejected by processing rules Section 5.2.2.1 case 1.

Implementations permitting multiple acceptable algorithms and/or truncations SHOULD permit this list to be ordered by presumed strength and SHOULD allow different truncations for the same algorithm to be treated as separate entities in this list. When so implemented, policies SHOULD accept a presumed stronger algorithm and truncation than the minimum strength required by the policy.

8. Shared Secrets

Secret keys are very sensitive information and all available steps should be taken to protect them on every host on which they are stored. Generally such hosts need to be physically protected. If they are multi-user machines, great care should be taken that unprivileged users have no access to keying material. Resolvers often run unprivileged, which means all users of a host would be able to see whatever configuration data is used by the resolver.

A name server usually runs privileged, which means its configuration data need not be visible to all users of the host. For this reason, a host that implements transaction-based authentication should probably be configured with a "stub resolver" and a local caching and forwarding name server. This presents a special problem for [RFC2136] which otherwise depends on clients to communicate only with a zone's authoritative name servers.

Use of strong random shared secrets is essential to the security of TSIG. See [RFC4086] for a discussion of this issue. The secret SHOULD be at least as long as the keyed hash output [RFC2104].

9. IANA Considerations

IANA maintains a registry of algorithm names to be used as "Algorithm Names" as defined in Section 4.2. Algorithm names are text strings encoded using the syntax of a domain name. There is no structure to the names, and algorithm names are compared as if they were DNS names, i.e., comparison is case insensitive. Previous specifications [RFC2845] and [RFC4635] defined values for the HMAC-MD5 and some HMAC-SHA algorithms. IANA has also registered "gss-tsig" as an identifier for TSIG authentication where the cryptographic operations are delegated to the Generic Security Service (GSS) [RFC3645]. This document adds to allowed algorithms, and the registry should be updated with the names listed in Table 2.

New algorithms are assigned using the IETF Review policy defined in [RFC8126]. The algorithm name HMAC-MD5.SIG-ALG.REG.INT looks like a fully-qualified domain name for historical reasons; other algorithm names are simple, single-component names.

IANA maintains a registry of RCODES (error codes), including "TSIG Error values" to be used for "Error" values as defined in Section 4.2. New error codes are assigned and specified as in [RFC6895].

10. Security Considerations

The approach specified here is computationally much less expensive than the signatures specified in DNSSEC. As long as the shared secret key is not compromised, strong authentication is provided between two DNS systems, e.g., for the last hop from a local name server to the user resolver, or between primary and secondary nameservers.

Recommendations for choosing and maintaining secret keys can be found in [RFC2104]. If the client host has been compromised, the server should suspend the use of all secrets known to that client. If possible, secrets should be stored in encrypted form. Secrets should never be transmitted in the clear over any network. This document does not address the issue on how to distribute secrets except that it mentions the possibilities of manual configuration and the use of TKEY [RFC2930]. Secrets SHOULD NOT be shared by more than two entities; any such additional sharing would allow any party knowing the key to impersonate any other such party to members of the group.

This mechanism does not authenticate source data, only its transmission between two parties who share some secret. The original source data can come from a compromised zone master or can be corrupted during transit from an authentic zone master to some "caching forwarder." However, if the server is faithfully performing the full DNSSEC security checks, then only security checked data will be available to the client.

A fudge value that is too large may leave the server open to replay attacks. A fudge value that is too small may cause failures if machines are not time synchronized or there are unexpected network delays. The RECOMMENDED value in most situations is 300 seconds.

To prevent cross-algorithm attacks, there SHOULD only be one algorithm associated with any given key name.

In several cases where errors are detected, an unsigned error message must be returned. This can allow for an attacker to spoof or manipulate these responses. Section 5.4 recommends logging these as errors and continuing to wait for a signed response until the request times out.

Although the strength of an algorithm determines its security, there have been some arguments that mild truncation can strengthen a MAC by reducing the information available to an attacker. However, excessive truncation clearly weakens authentication by reducing the number of bits an attacker has to try to break the authentication by brute force [RFC2104].

Significant progress has been made recently in cryptanalysis of hash functions of the types used here. While the results so far should not affect HMAC, the stronger SHA-256 algorithm is being made mandatory as a precaution.

See also the Security Considerations section of [RFC2104] from which the limits on truncation in this RFC were taken.

10.1. Issue Fixed in this Document

When signing a DNS reply message using TSIG, the MAC computation uses the request message's MAC as an input to cryptographically relate the reply to the request. The original TSIG specification [RFC2845] required that the TIME values be checked before the request's MAC. If the TIME was invalid, some implementations failed to carry out further checks and could use an invalid request MAC in the signed reply.

This document makes it a mandatory that the request MAC is considered to be invalid until it has been validated: until then, any answer must be unsigned. For this reason, the request MAC is now checked before the TIME value.

10.2. Why not DNSSEC?

These extracts from the original document [RFC2845] (updated to reference current standards) analyze DNSSEC in order to justify the introduction of TSIG.

DNS has recently been extended by DNSSEC ([RFC4033], [RFC4034] and [RFC4035]) to provide for data origin authentication, and public key distribution, all based on public key cryptography and public key based digital signatures. To be practical, this form of security generally requires extensive local caching of keys and tracing of authentication through multiple keys and signatures to a pre-trusted locally configured key.

One difficulty with the DNSSEC scheme is that common DNS implementations include simple "stub" resolvers which do not have caches. Such resolvers typically rely on a caching DNS server on another host. It is impractical for these stub resolvers to perform general DNSSEC authentication and they would naturally depend on their caching DNS server to perform such services for them. To do so securely requires secure communication of queries and responses. DNSSEC provides public key transaction signatures to support this, but such signatures are very expensive computationally to generate. In general, these require the same complex public key logic that is impractical for stubs.

and

A second area where use of straight DNSSEC public key based mechanisms may be impractical is authenticating dynamic update [RFC2136] requests. DNSSEC provides for request signatures but with DNSSEC they, like transaction signatures, require computationally expensive public key cryptography and complex authentication logic. Secure Domain Name System Dynamic Update ([RFC3007]) describes how different keys are used in dynamically updated zones.

11. References

11.1. Normative References

- [FIPS180-4] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, August 2015.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <<https://www.rfc-editor.org/info/rfc2845>>.
- [RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <<https://www.rfc-editor.org/info/rfc3597>>.
- [RFC4635] Eastlake 3rd, D., "HMAC SHA (Hashed Message Authentication Code, Secure Hash Algorithm) TSIG Algorithm Identifiers", RFC 4635, DOI 10.17487/RFC4635, August 2006, <<https://www.rfc-editor.org/info/rfc4635>>.

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

11.2. Informative References

- [BCP201] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015, <<https://www.rfc-editor.org/info/bcp201>>.
- [CVE-2017-11104] Common Vulnerabilities and Exposures, "CVE-2017-11104: Improper TSIG validity period check can allow TSIG forgery", June 2017, <<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-11104>>.
- [CVE-2017-3142] Common Vulnerabilities and Exposures, "CVE-2017-3142: An error in TSIG authentication can permit unauthorized zone transfers", June 2017, <<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-3142>>.
- [CVE-2017-3143] Common Vulnerabilities and Exposures, "CVE-2017-3143: An error in TSIG authentication can permit unauthorized dynamic updates", June 2017, <<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-3143>>.
- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <<https://www.rfc-editor.org/info/rfc1321>>.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<https://www.rfc-editor.org/info/rfc2136>>.
- [RFC2930] Eastlake 3rd, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, DOI 10.17487/RFC2930, September 2000, <<https://www.rfc-editor.org/info/rfc2930>>.

- [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <<https://www.rfc-editor.org/info/rfc3007>>.
- [RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <<https://www.rfc-editor.org/info/rfc3174>>.
- [RFC3645] Kwan, S., Garg, P., Gilroy, J., Esibov, L., Westhead, J., and R. Hall, "Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG)", RFC 3645, DOI 10.17487/RFC3645, October 2003, <<https://www.rfc-editor.org/info/rfc3645>>.
- [RFC3874] Housley, R., "A 224-bit One-way Hash Function: SHA-224", RFC 3874, DOI 10.17487/RFC3874, September 2004, <<https://www.rfc-editor.org/info/rfc3874>>.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, DOI 10.17487/RFC4868, May 2007, <<https://www.rfc-editor.org/info/rfc4868>>.
- [RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <<https://www.rfc-editor.org/info/rfc6151>>.

- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC6895] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", BCP 42, RFC 6895, DOI 10.17487/RFC6895, April 2013, <<https://www.rfc-editor.org/info/rfc6895>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [SHA1SHAMBLES]
Leurent, G. and T. Peyrin, "SHA-1 is a Shambles", January 2020, <<https://eprint.iacr.org/2020/014.pdf>>.

Appendix A. Acknowledgments

This document consolidates and updates the earlier documents by the authors of [RFC2845] (Paul Vixie, Olafur Gudmundsson, Donald E. Eastlake 3rd and Brian Wellington) and [RFC4635] (Donald E. Eastlake 3rd).

The security problem addressed by this document was reported by Clement Berthaux from Synacktiv.

Note for the RFC Editor (to be removed before publication): the first 'e' in Clement is a fact a small 'e' with acute, unicode code U+00E9. I do not know if xml2rfc supports non ASCII characters so I prefer to not experiment with it. BTW I am French too so I can help if you have questions like correct spelling...

Peter van Dijk, Benno Overeinder, Willem Toroop, Ondrej Sury, Mukund Sivaraman and Ralph Dolmans participated in the discussions that prompted this document. Mukund Sivaraman, Martin Hoffman and Tony Finch made extremely helpful suggestions concerning the structure and wording of the updated document.

Appendix B. Change History (to be removed before publication)

RFC EDITOR: Please remove this appendix before publication.

draft-dupont-dnsop-rfc2845bis-00

[RFC4635] was merged.

Authors of original documents were moved to Acknowledgments (Appendix A).

Section 2 was updated to [RFC8174] style.

Spit references into normative and informative references and updated them.

Added a text explaining why this document was written in the Abstract and at the beginning of the introduction.

Clarified the layout of TSIG RDATA.

Moved the text about using DNSSEC from the Introduction to the end of Security Considerations.

Added the security clarifications:

1. Emphasized that MAC is invalid until it is successfully validated.
2. Added requirement that a request MAC that has not been successfully validated MUST NOT be included into a response.
3. Added requirement that a request that has not been validated MUST NOT generate a signed response.
4. Added note about MAC too short for the local policy to Section 5.3.2.
5. Changed the order of server checks and swapped corresponding sections.
6. Removed the truncation size limit "also case" as it does not apply and added confusion.
7. Relocated the error provision for TSIG truncation to the new Section 5.2.4. Moved from RCODE 22 to RCODE 9 and TSIG ERROR 22, i.e., aligned with other TSIG error cases.
8. Added Section 5.4.4 about truncation error handling by clients.
9. Removed the limit to HMAC output in replies as a request which specified a MAC length longer than the HMAC output is invalid according to the first processing rule in Section 5.2.2.1.

10. Promoted the requirement that a secret length should be at least as long as the HMAC output to a SHOULD [RFC2119] key word.
11. Added a short text to explain the security issue.

draft-dupont-dnsop-rfc2845bis-01

Improved wording (post-publication comments).

Specialized and renamed the "TSIG on TCP connection" (Section 5.3.1) to "TSIG on zone transfer over a TCP connection". Added a SHOULD for a TSIG in each message (was envelope) for new implementations.

draft-ietf-dnsop-rfc2845bis-00

Adopted by the IETF DNSOP working group: title updated and version counter reset to 00.

draft-ietf-dnsop-rfc2845bis-01

Relationship between protocol change and principle of assuming the request MAC is invalid until validated clarified. (Jinmei Tatuya)

Cross reference to considerations for forwarding servers added. (Bob Harold)

Added text from [RFC3645] concerning the signing behavior if a secret key is added during a multi-message exchange.

Added reference to [RFC6895].

Many improvements in the wording.

Added RFC 2845 authors as co-authors of this document.

draft-ietf-dnsop-rfc2845bis-02

Added a recommendation to copy time fields in BADKEY errors. (Mark Andrews)

draft-ietf-dnsop-rfc2845bis-03

Further changes as a result of comments by Mukund Sivaraman.

Miscellaneous changes to wording.

draft-ietf-dnsop-rfc2845bis-04

Major restructuring as a result of comprehensive review by Martin Hoffman. Amongst the more significant changes:

- * More comprehensive introduction.
- * Merged "Protocol Description" and "Protocol Details" sections.
- * Reordered sections so as to follow message exchange through "client sending", "server receipt", "server sending", "client receipt".
- * Added miscellaneous clarifications.

draft-ietf-dnsop-rfc2845bis-05

Make implementation of HMAC-MD5 optional.

Require that the Fudge field in BADTIME response be equal to the Fudge field received from the client.

Added comment concerning the handling of BADTIME messages due to out of order packet reception.

draft-ietf-dnsop-rfc2845bis-06

Wording changes and minor corrections after feedback.

draft-ietf-dnsop-rfc2845bis-07

Updated text about use of hmac-sha1 using suggestion from Tony Finch.

Corrected name of review policy used for new algorithms.

draft-ietf-dnsop-rfc2845bis-08

Addressed comments from IESG review. These can be found at <https://datatracker.ietf.org/doc/draft-ietf-dnsop-rfc2845bis/ballot>. Significant changes are:

- * Added references to CVEs that initiated this draft.
- * Added reference to paper describing SHA1 collisions.
- * Modified some paragraphs to remove language that has not "aged well".

- * Mentioned that multiple keys allows for periodic key rotation.
- * Noted that TSIG detects interruption of packet sequence but not premature termination.
- * Added new algorithms to the algorithm list.
- * Marked hmac-sha224 as NOT RECOMMENDED.
- * Added recommendation that there should only be one algorithm for each key.
- * Added some paragraphs to the security recommendations section.

Other changes:

- * Explicitly define contents Error field in requests. State that "Other Data" currently has no meaning in requests.
- * Reworked the section on client processing of response to remove ambiguity.
- * Section on TSIG over TCP now mentions zone transfer as an example, rather than the entire section being about zone transfers.
- * Note that quote from RFC2845 in "What is DNSSEC?" section has been edited to refer to the latest standards.

draft-ietf-dnsop-rfc2845bis-09

Change use of hmac-224 from NOT RECOMMENDED to MAY.

Authors' Addresses

Francis Dupont
Internet Systems Consortium, Inc.
PO Box 360
Newmarket, NH 03857
United States of America

Email: Francis.Dupont@fdupont.fr

Stephen Morris
Internet Systems Consortium, Inc.
PO Box 360
Newmarket, NH 03857
United States of America

Email: sa.morris8@gmail.com

Paul Vixie
Farsight Security Inc
177 Bovet Road, Suite 180
San Mateo, CA 94402
United States of America

Email: paul@redbarn.org

Donald E. Eastlake 3rd
Futurewei Technologies
2386 Panoramic Circle
Apopka, FL 32703
United States of America

Email: d3e3e3@gmail.com

Olafur Gudmundsson
Cloudflare
San Francisco, CA 94107
United States of America

Email: olafur+ietf@cloudflare.com

Brian Wellington
Akamai
United States of America

Email: bwellington@akamai.com

Network Working Group
Internet-Draft
Updates: 7706 (if approved)
Intended status: Informational
Expires: December 26, 2018

W. Kumari
Google
P. Hoffman
ICANN
June 24, 2018

Decreasing Access Time to Root Servers by Running One On The Same Server
draft-kh-dnsop-7706bis-01

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.]

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, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Updates from RFC 7706	4
1.2. Requirements Notation	5
2. Requirements	5
3. Operation of the Root Zone on the Local Server	5
4. Using the Root Zone Server on the Same Host	7
5. Security Considerations	7
6. References	7
6.1. Normative References	7
6.2. Informative References	8
Appendix A. Current Sources of the Root Zone	8
Appendix B. Example Configurations of Common Implementations	9
B.1. Example Configuration: BIND 9.9	9
B.2. Example Configuration: Unbound 1.4 and NSD 4	10
B.3. Example Configuration: Microsoft Windows Server 2012	11
Acknowledgements	12
Authors' Addresses	12

1. Introduction

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, partially because the negative answers are cached for a much shorter period of time. A slow path between the recursive resolver and the closest root server has a negative effect on the resolver's customers.

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.

This document describes a method for the operator of a recursive resolver to greatly speed these queries and to hide them 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.

The primary goals of this design are to provide faster negative responses to stub resolver queries that contain queries that result in NXDOMAIN responses, and to prevent queries and responses from being visible on the network. This design will probably have little effect on getting faster positive 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.

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.

It is important to note that the design described in this document is controversial. There is not consensus on whether this is a "best practice". In fact, many people feel that it is an excessively risky practice because it introduces a new operational piece to local DNS operations where there was not one before. 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 requires the addition of authoritative name server software running on the same machine as the recursive resolver. Thus, recursive resolver software such as BIND will not need to add much new functionality, but recursive resolver software such as

Unbound will need to be able to talk to an authoritative server (such as NSD) running on the same host. However, more recursive resolver software might add the capabilities described in this document in the future.

A different approach to solving 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.

[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 draft.]

[Give a clearer comparison of software that allows slaving the root zone in the software (such as BIND) versus resolver software that requires a local slaved root zone (Unbound).]

[Add examples of other resolvers such as Knot Resolver and PowerDNS Recursor, and maybe Windows Server.]

[Add discussion of BIND slaving the root zone in the same view instead of using different views.]

[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.]

[Refresh list of where one can get copies of the root zone.]

[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:

- o The system MUST be able to validate a zone with DNSSEC [RFC4033].
- o The system MUST have an up-to-date copy of the key used to sign the DNS root.
- o The system MUST be able to retrieve a copy of the entire root zone (including all DNSSEC-related records).
- o 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 IPv4 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

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

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

6.2. Informative References

[Manning2013] Manning, W., "Client Based Naming", 2013, <http://www.sfc.wide.ad.jp/dissertation/bill_e.html>.

[RFC8198] Fujiwara, K., Kato, A., and W. Kumari, "Aggressive Use of DNSSEC-Validated Cache", RFC 8198, DOI 10.17487/RFC8198, July 2017, <<https://www.rfc-editor.org/info/rfc8198>>.

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:

- o b.root-servers.net
- o c.root-servers.net
- o f.root-servers.net
- o g.root-servers.net
- o 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 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.

The examples here use a loopback address of 127.12.12.12, but typical installations will use 127.0.0.1. The different address is used in order to emphasize that the root server does not need to be on the device at the name "localhost" which is often locally served as 127.0.0.1.

B.1. Example Configuration: BIND 9.9

BIND 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.

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

When slaving a zone, BIND 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 {
            192.228.79.201; # b.root-servers.net
            192.33.4.12;    # c.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
            2001:500:84::b; # b.root-servers.net
            2001:500:2f::f; # f.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 1.4 and NSD 4

Unbound and NSD are separate software packages. Because of this, there is no "fate-sharing" between the two servers in the following configurations. That is, if the root server instance (NSD) dies, the recursive resolver instance (Unbound) will probably keep running but will not be able to resolve any queries for the root zone. Therefore, the administrator of this configuration might want to carefully monitor the NSD instance and restart it immediately if it dies.

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

```
# Configuration for Unbound
server:
    do-not-query-localhost: no
stub-zone:
    name: "."
    stub-prime: no
    stub-addr: 127.12.12.12

# Configuration for NSD
server:
    ip-address: 127.12.12.12
zone:
    name: "."
    request-xfr: 192.228.79.201 NOKEY # b.root-servers.net
    request-xfr: 192.33.4.12 NOKEY   # c.root-servers.net
    request-xfr: 192.5.5.241 NOKEY   # f.root-servers.net
    request-xfr: 192.112.36.4 NOKEY  # g.root-servers.net
    request-xfr: 193.0.14.129 NOKEY  # k.root-servers.net
    request-xfr: 192.0.47.132 NOKEY  # xfr.cjr.dns.icann.org
    request-xfr: 192.0.32.132 NOKEY  # xfr.lax.dns.icann.org
    request-xfr: 2001:500:84::b NOKEY # b.root-servers.net
    request-xfr: 2001:500:2f::f NOKEY # f.root-servers.net
    request-xfr: 2001:7fd::1 NOKEY   # k.root-servers.net
    request-xfr: 2620:0:2830:202::132 NOKEY # xfr.cjr.dns.icann.org
    request-xfr: 2620:0:2d0:202::132 NOKEY # xfr.lax.dns.icann.org
```

B.3. Example Configuration: Microsoft Windows Server 2012

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 but to a very different problem (intermittent connectivity, instead of constant but slow connectivity) 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.

Authors' Addresses

Warren Kumari
Google

Email: Warren@kumari.net

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 9, 2019

L. Song
Beijing Internet Institute
S. Wang
Beijing Normal University
March 8, 2019

ATR: Additional Truncation Response for Large DNS Response
draft-song-atr-large-resp-03

Abstract

As the increasing use of DNSSEC and IPv6, there are more public evidence and concerns on IPv6 fragmentation issues due to larger DNS payloads over IPv6. This memo introduces an simple improvement on DNS server by replying an additional truncated response just after the normal fragmented response. It can be used to relieve users suffering on DNS latency and failures due to large DNS response. An ATR Experiment was done to show how well it works and some operational issues are discussed in this memo as well.

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 9, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. The ATR mechanism	3
3. Experiment on how well ATR works	5
4. Operational considerations	6
4.1. ATR timer	6
4.2. ATR payload size	7
4.3. Less aggressiveness of ATR	8
5. Security Considerations	8
6. IANA considerations	8
7. Acknowledgments	9
8. References	9
Appendix A. Considerations on Resolver awareness of ATR	11
Appendix B. Revision history of this document	11
B.1. draft-song-atr-large-resp-01	11
B.2. draft-song-atr-large-resp-02	12
B.3. draft-song-atr-large-resp-03	12
Authors' Addresses	13

1. Introduction

Large DNS response is identified as a issue for a long time. There is an inherent mechanism defined in [RFC1035] to handle large DNS response (larger than 512 octets) by indicating (set TrunCation bit) the resolver to fall back to query via TCP. Due to the fear of cost of TCP, EDNS(0) [RFC6891] was proposed which encourages server to response larger response instead of falling back to TCP. However, as the increasing use of DNSSEC and IPv6, there are more public evidence [DNSSEC-impact] and concerns on user's suffering due to packets dropping caused by IPv6 fragmentation in DNS due to large DNS response.

It is observed that some IPv6 network devices like firewalls intentionally choose to drop the IPv6 packets with fragmentation Headers [I-D.taylor-v6ops-fragdrop]. [RFC7872] reported more than 30% drop rates for sending fragmented packets. Regarding IPv6 fragmentation issue due to larger DNS payloads in response, one measurement [IPv6-frag-DNS] reported 35% of endpoints using IPv6-capable DNS resolver can not receive a fragmented IPv6 response over UDP. Depending on retry model, the resolver's failing to receive fragmented response may experience long latency or failure due to timeout and reties. And, most of the underlying issues with

fragments are unrevealed due to good redundancy and resilience of DNS and dual-stack network.

Generally speaking there are two approaches for this issue. One is to make the DNS response as small as possible, for example, using ECC instead of RSA to shorten the size of Key and signature. However, few zones are signed by ECC for the time being. In addition there is an uncertainty in the algorithm rollover from RSA to ECC. Another approach is to fall back to TCP by setting on either server side or client side. For resolver it is to set EDNS0 bufsize below a certain number. For authoritative servers it is to set their maximum UDP response size small enough.

However, one study [Not-speak-TCP] shows that about 17% of resolvers in the samples can not ask a query in TCP when they receive truncated response. It seems a dilemma to choose hurting either the users who can not receive fragments or the users without TCP fallback capacity. There is also some voice of "moving all DNS over TCP". But it is generally desired that DNS can keep the efficiency and high performance by using DNS UDP in most of time and fallback as soon as possible to TCP if necessary for some case.

To relieve the problem, this memo introduces a small improvement on DNS responding process by replying an Additional Truncated Response (ATR) just after a normal large response which is to be fragmented. It is a hybrid approach of using UDP when we can, and TCP only when we must. It does not require any changes on resolver and has a deploy-and-gain feature to encourage operators to implement it to benefit their resolvers.

[REMOVE BEFORE PUBLICATION] Note that ATR is not just a proposed idea. Some advocates of ATR implemented it based on BIND9 (https://gitlab.isc.org/isc-projects/bind9/merge_requests/158). And some verify it based on a large-scale experiment platform of APNIC lab Section 3 which is introduced in this memo.

2. The ATR mechanism

The ATR mechanism is very simple that it involves an ATR module in the responding process of current DNS implementation. As shown in the following diagram the ATR module is right after truncation loop if the packet is not going to be fragmented.

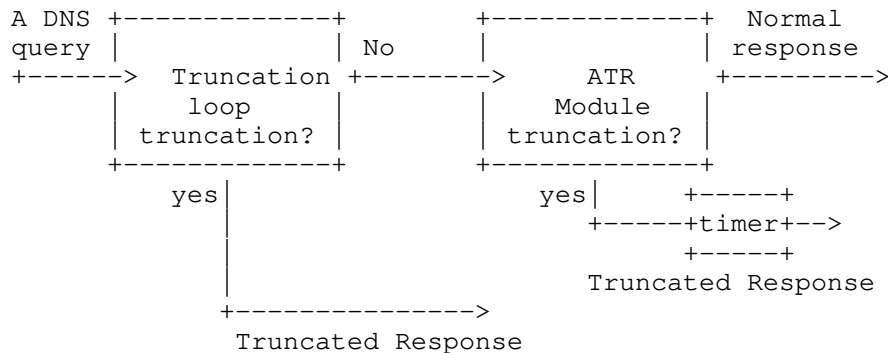


Figure 1: High-Level Testbed Components

The ATR responding process goes as follows:

- o When an authoritative server receives a query and enters the responding process, it first go through the normal truncation loop to see whether the size of response surpasses the EDNS0 payload size. If yes, it ends up with responding a truncated packets. If no, it enters the ATR module.
- o In ATR module, similar like truncation loop, the size of response is compared with a value called ATR payload size. If the response of a query is larger than ATR payload size, the server firstly sends the normal response and then coin a truncated response with the same ID of the query.
- o The server can reply the coined truncated response in no time. But considering the possible impact of network reordering, it is suggested a timer to delay the second truncated response, for example 10~50 millisecond which can be configured by local operation.

Note that the choice of ATR payload size and timer SHOULD be configured locally. And the operational consideration and guidance is discussed in Section 4.2 and Section 4.1 respectively.

There are three typical cases of ATR-unaware resolver behavior when a resolver send query to an ATR server in which the server will generate a large response with fragments:

- o Case 1: a resolver (or sub-resolver) will receive both the large response and a very small truncated response in sequence. It will happily accepts the first response and drop the second one because the transaction is over.

- o Case 2: In case a fragment is dropped in the middle, the resolver will end up with only receiving the small truncated response. It will retry using TCP in no time.
- o Case 3: For those (probably 30%*17% of them) who can not speak TCP and sitting behind a firewall stubbornly dropping fragments. Just say good luck to them!

In the case authoritative server truncated all response surpass certain value, for example setting IPv6-edns-size to 1220 octets, ATR will be helpful for resolver with TCP capacity, because the resolver still has a fair chance to receive the large response.

3. Experiment on how well ATR works

It is worth mentioning APNIC report [How-ATR-Work] on "How well does ATR actually work?" done by Geoff Huston and Joao Damas after 00 version of ATR draft. It was reported firstly in IEPG meeting before IETF 101 and then posted in APNIC Blog later.

It is said the test was performed over 55 million endpoints, using an on-line ad distribution network to deliver the test script across the Internet. The result is positive that ATR works! From the end users' perspective, in some 9% of IPv4 cases the use of ATR by the server will improve the speed of resolution of a fragmented UDP response by signaling to the client an immediate switch to TCP to perform a re-query. The IPv6 behavior would improve the resolution times in 15% of cases.

It also analyzed the pros and cons of ATR. On one hand, It is said that ATR certainly looks attractive if the objective is to improve the speed of DNS resolution when passing large DNS responses. And ATR is incrementally deployable in favor of decision made by each server operator. On another hand, ATR also has some negative or unanswered factors. One is adding another DNS DDoS attack vector due to the additional packet sent by ATR, (author's note: very small adding actually.) Another issue is risk of RO by the choice of the delay timer which is discussed fully in Section 4.1. It is also founded that the trailing UDP packet may generate ICMP Port Unreachable messages back to the server as a kind of noise (a rate of approximately 1 in 5 responses in our experiments). Note that in author's argument, it is not a big issue and the server can simply ignore it if it decides to adopt ATR.

As a conclusion, it is said that "ATR does not completely fix the large response issue. If a resolver cannot receive fragmented UDP responses and cannot use TCP to perform DNS queries, then ATR is not going to help. But where there are issues with IP fragment

filtering, ATR can make the inevitable shift of the query to TCP a lot faster than it is today. But it does so at a cost of additional packets and additional DNS functionality". "If a faster DNS service is your highest priority, then ATR is worth considering", said at the end of this report

4. Operational considerations

There are some operational consideration on ATR, such as the parameter of the ATR timer and ATR payload size, and policies on when ATR is triggered to avoid side-effect.

4.1. ATR timer

As introduced in Section 2 ATR timer is a way to avoid the impact of network reordering (RO). The value of the timer is critical, because if the delay is too short, the ATR response may be received earlier than the fragmented response (the first piece), the resolver will fall back to TCP bearing the cost which should have been avoided. If the delay is too long, the client may timeout and retry which negates the incremental benefit of ATR. Generally speaking, the delay of the timer should be "long enough, but not too long".

To the best knowledge of author, the nature of RO is characterized as follows hopefully helping ATR users understand RO and how to operate ATR appropriately in RO context.

- o RO is mainly caused by the parallelism in Internet components and links other than network anomaly [Bennett]. It was observed that RO is highly related to the traffic load of Internet components. So RO will long exists as long as the traffic load continue increase and the parallelism is used to enhance network throughput.
- o The probability of RO varies largely depending on the different tests samples. Some work shown RO probability below 2% [Paxson] [Tinta] and another work was above 90% [Bennett]. But it is agreed that RO is site-dependent and path-dependent. It is observed in that when RO happens, it is mostly exhibited consistently in a small percentages of the paths. It is also observed that higher rates smaller packets were more prone to RO because the sending inter-spacing time was small.
- o It was reported that the inter-arrival time of RO varies from a few milliseconds to multiple tens of milliseconds [Tinta]. And the larger the packet the larger the inter-arrival time, since larger packets will take longer to be transmitted.

Reasonably we can infer that firstly RO should be taken into account because it long exists due to middle Internet components which can not be avoided by end-to-end way. Secondly the mixture of larger and small packets in ATR case will increase the inter-arrival time of RO as well as the its probability. The good news is that the RO is highly site specific and path specific, and persistent which means the ATR operator is able to identify a few sites and paths, setup a tunable timer setting for them, or just put them into a blacklist without replying ATR response.

Based on the above analysis it is hard to provide a perfect value of ATR timer for all ATR users due to the diversity of networks. It seems OK to set the timer with a range from ten to hundreds ms, just below the timeout setting of typical resolver. It is suggested that a decision should be made as operator-specific according to the statistic of the RTT of their users. Some measurement shown [Brownlee][Liang] the mean of response time is below 50 ms for the sites with lots of anycast instance like L-root, .com and .net name servers. For that sites, delay less than 50 ms is appropriate.

4.2. ATR payload size

Regarding the operational choice for ATR payload size, there are some good input from APNIC study [scoring-dns-root] on how to react to large DNS payload for authoritative server. The difference in ATR is that ATR focuses on the second response after the ordinary response.

For IPv4 DNS server, it is suggested the study that do not truncate and fragment IPv4 UDP response with a payload up to 1472 octets which is Ethernet MTU(1500) minus the sum of IPv4 header(20) and UDP header(8). The reason is to avoid gratuitously fragmenting outbound packets and TCP fallback at the source.

In the case of ATR, the first ordinary response is emitted without knowing it be to fragmented or not on the path. If a large value is set up to 1472 octets, payload size between 512 octets and the large value size will probably get fragmented by aggressive firewalls which leads losing the benefit of ATR. If ATR payload size set exactly 512 octets, in most of case ATR response and the single unfragmented packets are under a race at the risk of RO.

Given IPv4 fragmentation issue is not so serious compared to IPv6, it is suggested in this memo to set ATR payload size 1472 octets which means ATR only fit large DNS response larger than 1500 octets in IPv4.

For IPv6 DNS server, similar to IPv4, the APNIC study is suggested that do not truncate IPv6 UDP packets with a payload up to 1,452

octets which is Ethernet MTU(1500) minus the sum of IPv6 header(40) and UDP header(8). 1452 octets is chosen to avoid TCP fallback in the context that most TCP MSS in the root server is not set probably at that time.

In the case of ATR considering the second truncated response, a smaller size: 1232 octets, which is IPv6 MTU for most network devices(1280) minus the sum of IPv6 header(40) and UDP header(8), should be chosen as ATR payload size to trigger necessary TCP fallback. As a complementary requirement with ATR, the TCP MSS should be set 1220 octets to avoid Packet Too Big ICMP message as suggested in the APNIC study.

In short, it is recommended that in IPv4 ATR payload size SHOULD be 1472 octets, and in IPv6 the value SHOULD be 1232 octets.

4.3. Less aggressiveness of ATR

There is a concern ATR sends TC=1 response too aggressively especially in the beginning of adoption. ATR can be implemented as an optional and configurable feature at the disposal of authoritative server operator. One of the idea to mitigate this aggressiveness, ATR may respond TC=1 responses at a low possibility, such as 10%.

Another way is to reply ATR response selectively. It is observed that RO and IPv6 fragmentation issues are path specific and persistent due to the Internet components and middle box. So it is reasonable to keep a ATR "whitelist" by counting the retries and recording the IP destination address of that large response causing many retries. ATR only acts to those queries from the IP address in the white list.

5. Security Considerations

There may be concerns on DDoS attack problem due to the fact that the ATR introduces multiple responses from authoritative server. The extra packet is pretty small. In the worst case, it's 50% more packets and they are small

DNS cookies [RFC7873] and RRL on authoritative may be possible solutions

6. IANA considerations

No IANA considerations for this memo

7. Acknowledgments

Many thanks to reviewers and their comments. Geoff Huston and Joao Damas did a testing on the question "How well does ATR actually work?". Alexander Dupuy proposed the idea to distinguish ATR responses from normal ones. Akira Kato contributed ideas on operational consideration. Shane Kerr help author with the security consideration. Stephane Bortzmeyer gave thought of happyeyeballs on resolver side.

Acknowledgments are also give to Mukund Sivaraman, Evan Hunt and Mark Andrews who implement it and maintained it in a brunch in BIND9 code base.

8. References

[ATR-Github]

"XML source file and test script of DNS ATR", September 2017, <https://github.com/songlinjian/DNS_ATR>.

[Bennett] Bennett, J. C. R., "Packet Reordering is Not Pathological Network Behavior", December 1999, <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.461.7629&rep=rep1&type=pdf>>.

[Brownlee]

Brownlee, N., "Response time distributions for global name servers", 2002, <<http://www.caida.org/publications/papers/2002/nsrtd/nsrtd.pdf>>.

[DNSSEC-impact]

Broek, G. V. D., "DNSSEC meets real world: dealing with unreachability caused by fragmentation", April 2014, <<https://repository.ubn.ru.nl/bitstream/handle/2066/132796/132796.pdf?sequence=1>>.

[How-ATR-Work]

Huston, G., "How well does ATR actually work?", April 2018, <<https://blog.apnic.net/2018/04/16/how-well-does-atr-actually-work/>>.

[I-D.taylor-v6ops-fragdrop]

Jaeggli, J., Colitti, L., Kumari, W., Vyncke, E., Kaeo, M., and T. Taylor, "Why Operators Filter Fragments and What It Implies", draft-taylor-v6ops-fragdrop-02 (work in progress), December 2013.

- [IPv6-frag-DNS] Huston, G., "Dealing with IPv6 fragmentation in the DNS", August 2017, <<https://blog.apnic.net/2017/08/22/dealing-ipv6-fragmentation-dns>>.
- [Liang] Liang, J., "Measuring Query Latency of Top Level DNS Servers", February 2013, <<https://netsec.ccert.edu.cn/duanhx/files/2013/02/latency.pdf>>.
- [Not-speak-TCP] Huston, G., "A Question of DNS Protocols", August 2013, <<https://labs.ripe.net/Members/gih/a-question-of-dns-protocols>>.
- [Paxson] Paxson, V., "End-to-End Internet Packet Dynamics", August 1999, <<https://cseweb.ucsd.edu/classes/fa01/cse222/papers/paxson-e2e-packets-sigcomm97.pdf>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/info/rfc6891>>.
- [RFC7872] Gont, F., Linkova, J., Chown, T., and W. Liu, "Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World", RFC 7872, DOI 10.17487/RFC7872, June 2016, <<https://www.rfc-editor.org/info/rfc7872>>.
- [RFC7873] Eastlake 3rd, D. and M. Andrews, "Domain Name System (DNS) Cookies", RFC 7873, DOI 10.17487/RFC7873, May 2016, <<https://www.rfc-editor.org/info/rfc7873>>.
- [scoring-dns-root] Huston, G., "Scoring the DNS Root Server System", November 2016, <<https://blog.apnic.net/2016/11/15/scoring-dns-root-server-system/>>.
- [Tinta] Tinta, S. P., "Characterizing End-to-End Packet Reordering with UDP Traffic", August 2009, <<https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/35247.pdf>>.

Appendix A. Considerations on Resolver awareness of ATR

ATR proposed in this memo is a server-side function which requires no change in resolver, so it is not required that resolver MUST recognize ATR and react accordingly. But it may be helpful for some cases where a resolver is able to recognize ATR response, for example by checking the large edns0 payload size and Truncation bit.

One case where ATR is used as a troubleshooting tool by which resolver operators are able to flag problematic name servers. The resolver operator is enabled to log cases where ATR responses are received without a (reassembled) UDP response to a query. In the case of receiving a ATR, RDNS can choose to restrict maximum EDNS to a lower value than the default 4096 that is currently used.

Another case is that when receiving a ATR response a ATR-aware resolver can adopt a "happy eyeballs" strategy by opening a separate transaction sending the query via TCP instead of falling back to TCP and closing the original UDP transaction. Listening to port 53 on both TCP and UDP will enhance the availability and reduce the latency. It will add more tolerance to network reordering issues as well. However, it should be taken into account about the balance of resolver's resource. Less priority should be given to that function when the resolver is "busy".

The awareness of ATR on resolver can also avoid sending ICMP Port Unreachable messages back to the server. In some implementations, reusing the same UDP sockets for multiple queries will not generate that ICMP noise.

However, resolver use case of ATR is currently outside of the scope of server-ATR proposal. It needs further discussion.

Appendix B. Revision history of this document

B.1. draft-song-atr-large-resp-01

After receiving reviews and comments, changes of 01 version are shown as follows:

- o Rewrite introduction and add another goal of ATR as a measuring tool;
- o Add section 3 indicating a ATR response. A bit in the EDNS0 OPT header is defined as an indicator of ATR response. The flag bit is called "ATR Response" (AT) bit;

- o Add Section 4 Operation considerations, which discuss ATR timer , ATR payload size, and less aggressiveness of ATR;
- o Add IANA consideration to register the AT bit;
- o Add section 7 Acknowledgments;
- o Append a list of references regarding Network reordering, and APNIC's study on IPv6 and DNS;
- o Add Appendix A, An introduce of APNIC testing work and author's comments;
- o Appendix B. Considerations on Resolver awareness of ATR;
- o Change the category="std" . It is said in RFC6891 IETF Standards Action is required for assignments of new EDNS(0) flags. So the draft should be categorized as standard track if registering AT bit is desired in this document.

Change history is also available in the public GitHub repository where this document is maintained: <https://github.com/songlinjian/DNS_ATR>.

B.2. draft-song-atr-large-resp-02

Changes in 02 version of ATR draft:

- o Remove the section of introduction of AT bit as well as requirement of IANA registration of that bit;
- o Change the category of this document to experimental and move the introduction of APNIC's experiment from Appendix A to section 3;
- o Add more names in Acknowledgments part after IETF102;

B.3. draft-song-atr-large-resp-03

Changes in 03 version of ATR draft:

- o Add related work in the introduction session;
- o Introduce ICMP noise as a finding of APNIC's experiment and propose how to avoid it in Appendix A;
- o Change the category from "exp" to "info";
- o Move to ISE for review.

- o Add one author S. Wang

Authors' Addresses

Linjian Song
Beijing Internet Institute
2nd Floor, Building 5, No.58 Jing Hai Wu Lu, BDA
Beijing 100176
P. R. China

Email: songlinjian@gmail.com
URI: <http://www.biigroup.com/>

Shengling Wang
Beijing Normal University
Beijing Normal University, No. 19, XinJieKouWai St., HaiDian District
Beijing 100875
P. R. China

Email: wangshengling@bnu.edu.cn
URI: <https://cist.bnu.edu.cn/>

DNSOP
Internet-Draft
Intended status: Standards Track Riphah Institute of Systems Engineering
Expires: November 12, 2018

T. Saraj, Ed.
M. Yousaf
A. Qayyum
Capital University of Science and Technology
May 11, 2018

IVIPTR: Resource Record for DNS
draft-tariq-dnsop-iviptr-01

Abstract

This document proposes a new DNS Resource Record IVIPTR which provides the capability to resolve the IPv4 address to IPv6 address and IPv6 address to IPv4 address. This document assumes that the reader is familiar with all the concepts and details discussed in Domain Names Concepts and Facilities [RFC1034] , Domain Names - Implementation and Specification [RFC1035]

Status of This Memo

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

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

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

This Internet-Draft will expire on November 12, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Motivation and Usecases	4
2.1. Usecase-01: Firewall Automation	4
2.2. Usecase-02: Promoting IPv6 Usage	5
2.3. Usecase-03: Customized Debugging Utilities	5
2.4. Usecase-04: Spam Filtering	5
3. The IVIPTR Resource Record	5
3.1. Ideal Scenario	6
3.2. Non-Ideal Scenario	6
3.3. Reverse zone file for IPv4 network prefix	7
3.4. Reverse zone file for IPv6 network prefix	7
4. Query Processing	7
4.1. Client Query: Case-01	9
4.2. Client Query: Case-02	9
5. Security Considerations	10
6. Acknowledgements	10
7. IANA Considerations	10
8. Normative References	10
Authors' Addresses	10

1. Introduction

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].

The current DNS standard does not support to resolve IPv4 address to IPv6 address and IPv6 address to IPv4 address. For example, if a user program initiate a query for AAAA resource record against an IPv4 address of a domain, the current DNS will return a negative answer normally with RCODE(3)-Non-Existent Domain. Using the current DNS standard, a user program can resolve IPv6 address for a desired IPv4 address by the process as in figure-01:

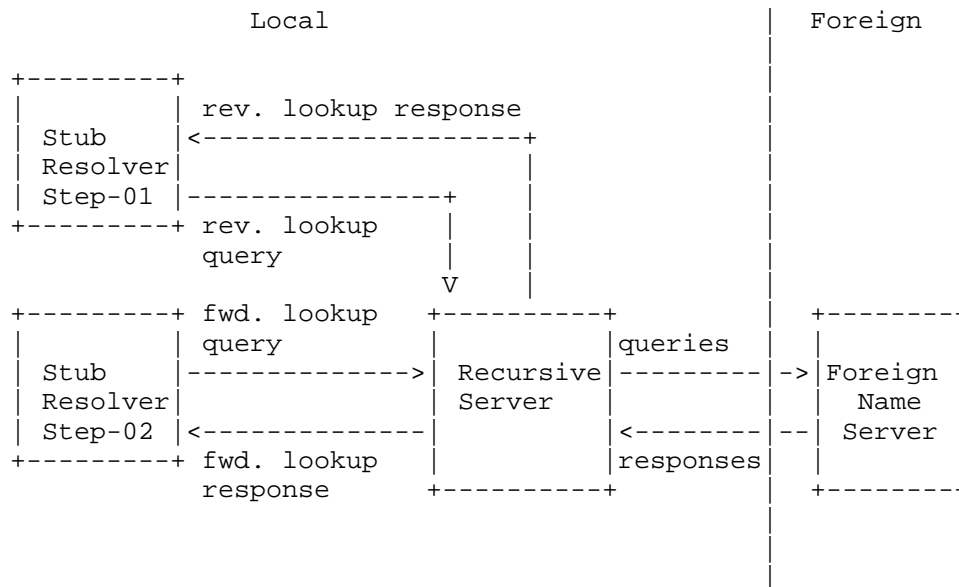


Figure 1

1. The stub-resolver in Step-01, sends a reverse lookup query for an A record to the recursive server to resolve the corresponding fully qualified domain name from the Foreign Name Server
2. The Foreign Name Server returns the PTR resource record against the query to the recursive server, which is responded back to the Stub-resolver as a response.
3. For the received domain name in Step-01, the stub-resolver in Step-02, sends a forward lookup query to recursive server to resolve the corresponding AAAA resource record from the Foreign Name Server.
4. The Foreign Name Server returns the AAAA resource record against the query to the recursive server, which is responded back to the Stub-resolver as a response.

Here, the bottleneck in this process is that now a days, mostly domains has different PTR records for a corresponding A or AAAA resource record. In this case the aforementioned process in figure-01 is not suitable. Also, this process requires to make changes to the Stub-resolver functionality to pursue the aforementioned process. Even, if the Stub-resolver functionality is modified it will work only if a single domain name is used for both A and AAAA record. The proposed solution (IVI PTR) is that, when the

Stub-resolver send a query to the recursive server for resolving AAAA record against an IPv4 address and vice versa, it will respond with the desired resource record (RR) without depending upon a Fully Qualified Domain Name(FQDN) knowledge on Stub-resolver. The term IVI in the proposed IVIPTR resource record is borrowed from one of the IPv4/IPv6 transition mechanisms address translation algorithm [RFC6219].

2. Motivation and Usecases

IPv4 is the principal protocol being used for communication in most of the organizations. Primarily, the need of IVIPTR RR in DNS evolved in a lab environment during the translation of IPv4 security rules to IPv6 security rules in a network security component (Firewall). This section discuss four usecases for the proposed DNS resource record.

2.1. Usecase-01: Firewall Automation

In network security components, mostly traffic monitoring is done through rule based filtering. An organization may enable IPv6 for certain reasons such as:

1. Functionality testing of a newly developed application with IPv6.
2. Performance and compatibility testing of application with IPv6.
3. Or, the organization has decided to keep their network on dual stack from onwards for transition purpose etc.

As a result the security guys has to maintain dual security rules for both Inbound and Outbound network traffic. This can be done by manually configuring the security rules in all network security components for the newly enabled Internet protocol IPv6. Mistakenly, configuring any security rule can result in an undesired consequences.

To automate the security configuration process in a network, there is a need to resolve IPv6 address for a corresponding IPv4 address against every security rule in a network security component (Firewall). The only resource in any network available for this automation process is the DNS. Currently in DNS, there is no such mechanism that can return IPv6 address of a domain if IPv4 address is known or vice versa. The IVIPTR Resource Record conceived as a solution to the problem for resolving IPv6 address if IPv4 address is known or IPv4 address if IPv6 address is known.

There may exist IPv4 or IPv6 address in network security components rules, which does not belong to any fully qualified domain name (FQDN) and thus, are out of the scope of this work. The presence of this IVIPTR Resource Record in the reverse zone file of an authoritative name server can result in automating a number of service for enabling them to reconfigure their security rules for the newly enabled address family protocol i.e. IPv4 or IPv6.

2.2. Usecase-02: Promoting IPv6 Usage

When accessing service such as FTP for a domain say example.com, a user can connect to the server by either:

1. ftp example.com
2. Or, ftp 192.168.0.1

For the second FTP access mechanism, the IVIPTR RR will help to retrieve the IPv6 address against the IPv4 address of the FTP server. Further, the user application will use the newly retrieved IPv6 for connectivity instead of the given one to promote the usage of IPv6 as the priority Internet address for connectivity.

2.3. Usecase-03: Customized Debugging Utilities

Debugging utilities such as traceroute can be customized in such a way that it will give detailed response. For example if a user gives a traceroute command as:

```
traceroute++ 192.168.0.1 or traceroute++ example.com
```

Thus, the output will be both PTR record and IVIPTR record.

2.4. Usecase-04: Spam Filtering

When applying spam filtering policy for a mail server such as mail.example.com, the IVIPTR can be helpful in providing additional details such as:

If filtering is performed on IPv4 address, the same can be done for IPv6 address for the corresponding mail server

3. The IVIPTR Resource Record

The IVIPTR RR has mnemonic IVIPTR and type code TBD (decimal). The IVIPTR RR has the following format:

```
<OWNER> <TTL> <CLASS> IVIPTR <IVI target >
```

The OWNER is either unqualified or fully qualified domain name depending upon the configuration of reverse zone file optional directive \$ORIGIN. The TTL and CLASS fields are the same as for all other PTR records in the reverse zone file. As for the usecases discussed in the previous section the fact of IVIPTR RR usage, it is to be believed that this resource record will not be required to access frequently or in some cases just once, so one can set a smaller TTL value for this resource record to facilitate the recursive server by reducing the cache from unnecessary increase.

IVIPTR is the new RR type that points to a fully qualified domain name (FQDN) i.e. IVI target in a reverse zone file. The <IVI target> from onwards for simplicity written as <target> SHOULD be a fully qualified domain name (FQDN).

The presence of <IVIPTR RR> in a reverse zone can be elaborate by considering the domain example.com. Realistically, most of the times labels in a domain name for an IPv4 and IPv6 glue record are different. There are two possible scenarios for configuration of forward lookup zone file.

3.1. Ideal Scenario

An ideal scenario for a forward lookup zone file would be the one in which, labels in a domain name are same for both IPv4 and IPv6 glue records as:

```
; zone file for example.com

x.example.com.  IN A 192.168.0.1

x.example.com.  IN AAAA 2001:DB8:0::1
```

3.2. Non-Ideal Scenario

A non-ideal scenario for a forward lookup zone file would be the one in which, labels in a domain name are slightly different for both IPv4 and IPv6 glue records as:

```
; zone file for example.com

x.example.com.  IN A 192.168.0.1

x6.example.com. IN AAAA 2001:DB8:0::1
```

The use of IVIPTR RR is effective only against forward lookup zone file Non-Ideal configuration scenario. Although, it will cause no issue with the Ideal scenario except additional processing overhead.

The representation of IIVIPTTR RR against both the IPv4 and IPv6 addresses would be as discussed in the next two sub-sections respectively.

3.3. Reverse zone file for IPv4 network prefix

When configuring a reverse zone file for example.com of IPv4 network prefix, the representation of IVIPTR RR type would be as:

```
; reverse zone file example.com for IPv4
1.0.168.192.IN-ADDR.ARPA.  IN PTR x.example.com.
1.0.168.192.IN-ADDR.ARPA.  IN IVIPTR x6.example.com.
```

3.4. Reverse zone file for IPv6 network prefix

When configuring a reverse zone file for example.com of IPv6 network prefix, the representation of IVIPTR RR type would be as:

```
; reverse zone file example.com for IPv6  
  
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.IP  
6.ARPA.    IN PTR x6.example.com.  
  
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.IP  
6.ARPA.    IN IVIPTR x.example.com.
```

For the purpose of simplicity in the forward lookup zone file, there is no referral RR Type such as CNAME is listed. In case of presence of any referral record in the forward lookup zone file the <IVI target > in both of the reverse zone files SHOULD be the same as the CNAME < target > in the forward lookup zone file. Thus, IVIPTR MUST follow the rule of robustness principle discussed in section 3.6.2 of [RFC1034] to avoid extra indirections in accessing information.

4. Query Processing

The IVIPTR follow the top level RR format and semantics as defined in the section 3.2.1 of RFC 1035 [RFC1035].

```

      0  1  2  3  4  5  6  7  8  9  0  1  1  1  1  1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|
|
|      NAME = 1.0.168.192.IN-ADDR.ARP.A.
|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|                                TYPE = IVPTR
|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|                                CLASS = IN
|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|                                TTL
|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|                                RDLENGTH
|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|                                RDATA
|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Figure 2

Where:

NAME: the owner name, same as in any reverse lookup query.

TYPE: the two octets field containing the IVIPTR RR TYPE code.

CLASS: two octets containing the RR IN CLASS code value 1.

TTL: the time interval in seconds that the resource record may be cached before the source of the information again to be contacted.

RDLENGTH: specifies the length of RDATA field.

RDATA: A variable length string of octets that represents the <IVI target> resource. The resource depends on the owner in the NAME field of the query.

The query processing is same as any other DNS query except that when the recursive server receives the response for the IVIPTR RR, first it will cache the response like any other resource record and then it will form a new query based on the rules in the sub-sections of this section.

4.1. Client Query: Case-01

If the original query NAME field contains IPv4 representation and TYPE field is IVIPTR then:

1. Upon receiving the response at the recursive server, it SHOULD form a new query.
2. The NAME field of the new query SHOULD be mapped appropriately in the desired format to the IVIPTR target in RDATA resource.
3. The TYPE field for the new query SHOULD be AAAA.
4. This query will be resolved as any other forward lookup query. Upon receiving the response which will contain AAAA RR type target, the recursive server will place this in the answer section of the original query request from client. The IVIPTR RR SHOULD cause no additional section processing.
5. In case of failure or any error the standard error response will be send back to the stub-resolver against the original query request.

4.2. Client Query: Case-02

If the original query NAME field contains IPv6 representation and TYPE field is IVIPTR then:

1. Upon receiving the response at the recursive server, it SHOULD form a new query.
2. The NAME field of the new query SHOULD be mapped appropriately in the desired format to the target in RDATA resource.
3. The TYPE field for the new query SHOULD be A.
4. This query will be resolved as any other forward lookup query. Upon receiving the response which will contain A RR type target, the recursive server will place this in the answer section of the original query request from client. The IVIPTR RR SHOULD cause no additional section processing.
5. In case of failure or any error the standard error response will be send back to the stub-resolver against the original query request.

5. Security Considerations

On a security-aware name server, while resolving the IVIPTTR the query processing involves a forward lookup on recursive server in both Section 4.1 and section 4.2 when the new query is formed. The forward lookup in both the cases SHOULD comply completely with the DNSSEC on a security-aware name server and stub-resolver.

6. Acknowledgements

7. IANA Considerations

8. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6219] Li, X., Bao, C., Chen, M., Zhang, H., and J. Wu, "The China Education and Research Network (CERNET) IVI Translation Design and Deployment for the IPv4/IPv6 Coexistence and Transition", RFC 6219, DOI 10.17487/RFC6219, May 2011, <<https://www.rfc-editor.org/info/rfc6219>>.

Authors' Addresses

Tariq Saraj (editor)
Riphah Institute of Systems Engineering
Aga Khan Road, Sector F-5/1
Islamabad, Federal Capital 44000
Pakistan

Phone: 00923345755556
Email: tariqsaraj@gmail.com

Muhammad Yousaf
Riphah Institute of Systems Engineering
Aga Khan Road, Sector F-5/1
Islamabad, Federal Capital 44000
Pakistan

Email: Muhammad.Yousaf@riu.edu.pk

Amir Qayyum
Capital University of Science and Technology
Kahuta Road, Islamabad Expressway
Islamabad, Federal Capital 44000
Pakistan

Email: aq@cust.edu.pk

Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: August 17, 2019

D. Wessels
P. Barber
M. Weinberg
Verisign
W. Kumari
Google
W. Hardaker
USC/ISI
February 13, 2019

Message Digest for DNS Zones
draft-wessels-dns-zone-digest-06

Abstract

This document describes an experimental protocol and new DNS Resource Record that can be used to provide a message digest over DNS zone data. The ZONEMD Resource Record conveys the message digest data in the zone itself. When a zone publisher includes an ZONEMD record, recipients can verify the zone contents for accuracy and completeness. This provides assurance that received zone data matches published data, regardless of how the zone data has been transmitted and received.

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

As specified at this time, ZONEMD is not designed for use in large, dynamic zones due to the time and resources required for digest calculation. The ZONEMD record described in this document includes fields reserved for future work to support large, dynamic zones.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 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	3
1.1. Motivation	4
1.2. Design Overview	5
1.3. Use Cases	6
1.3.1. Root Zone	6
1.3.2. Providers, Secondaries, and Anycast	6
1.3.3. Response Policy Zones	7
1.3.4. Centralized Zone Data Service	7
1.3.5. General Purpose Comparison Check	7
1.4. Requirements Language	7
2. The ZONEMD Resource Record	7
2.1. ZONEMD RDATA Wire Format	8
2.1.1. The Serial Field	8
2.1.2. The Digest Type Field	8
2.1.3. The Reserved Field	8
2.1.4. The Digest Field	8
2.2. ZONEMD Presentation Format	9
2.3. ZONEMD Example	9
3. Calculating the Digest	9
3.1. Canonical Format and Ordering	9
3.1.1. Order of RRsets Having the Same Owner Name	9
3.1.2. Duplicate RRs	10
3.2. Add ZONEMD Placeholder	10
3.3. Optionally Sign the Zone	10
3.4. Calculate the Digest	10
3.4.1. Inclusion/Exclusion Rules	11

3.5. Update ZONEMD RR	11
4. Verifying Zone Message Digest	11
5. Scope of Experimentation	13
6. IANA Considerations	13
6.1. ZONEMD RRtype	13
6.2. ZONEMD Digest Type	14
7. Security Considerations	14
7.1. Attacks Against the Zone Digest	14
7.2. Attacks Utilizing the Zone Digest	14
8. Privacy Considerations	15
9. Acknowledgments	15
10. Implementation Status	15
10.1. Authors' Implementation	15
10.2. Shane Kerr's Implementation	15
11. Change Log	16
12. References	18
12.1. Normative References	18
12.2. Informative References	19
Appendix A. Example Zones With Digests	21
A.1. Simple EXAMPLE Zone	21
A.2. Complex EXAMPLE Zone	21
A.3. The URI.ARPA Zone	22
A.4. The ROOT-SERVERS.NET Zone	25
Authors' Addresses	27

1. Introduction

In the DNS, a zone is the collection of authoritative resource records (RRs) sharing a common origin ([RFC7719]). Zones are often stored as files on disk in the so-called master file format [RFC1034]. Zones are generally distributed among name servers using the AXFR [RFC5936], and IXFR [RFC1995] protocols. Zone files can also be distributed outside of the DNS, with such protocols as FTP, HTTP, rsync, and even via email. Currently there is no standard way to verify the authenticity of a stand-alone zone.

This document introduces a new RR type that serves as a cryptographic message digest of the data in a zone. It allows a receiver of the zone to verify the zone's authenticity, especially when used in combination with DNSSEC. This technique makes the message digest a part of the zone itself, allowing verification the zone as a whole, no matter how it is transmitted. Furthermore, the digest is based on the wire format of zone data. Thus, it is independent of presentation format, such as changes in whitespace, capitalization, and comments.

DNSSEC provides three strong security guarantees relevant to this protocol:

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

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

1.1. Motivation

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

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

DNS Request and Transaction Signatures (SIG(0) [RFC2931]) is another protocol extension designed to authenticate individual DNS transactions. Whereas SIG records were originally designed to cover specific RR types, SIG(0) is used to sign an entire DNS message. Unlike TSIG, SIG(0) uses public key cryptography rather than shared secrets.

The Transport Layer Security protocol suite is also designed to provide channel security. One can easily imagine the distribution of zones over HTTPS-enabled web servers, as well as DNS-over-HTTPS [dns-over-https], and perhaps even a future version of DNS-over-TLS ([RFC7858]).

Unfortunately, the protections provided by these channel security techniques are (in practice) ephemeral and are not retained after the data transfer is complete. They can ensure that the client receives the data from the expected server, and that the data sent by the server is not modified during transmission. However, they do not guarantee that the server transmits the data as originally published, and do not provide any methods to verify data that is read after transmission is complete. For example, a name server loading saved zone data upon restart cannot guarantee that the on-disk data has not

been modified. For these reasons, it is preferable to secure the data itself.

Why not simply rely on DNSSEC, which provides certain data security guarantees? Certainly for zones that are signed, a recipient could validate all of the signed RRSets. Additionally, denial-of-existence records can prove that RRSets have not been added or removed. However, not all RRSets in a zone are signed. The design of DNSSEC stipulates that delegations (non-apex NS records) are not signed, and neither are any glue records. Thus, changes to delegation and glue records cannot be detected by DNSSEC alone. Furthermore, zones that employ NSEC3 with opt-out are susceptible to the removal or addition of names between the signed nodes. Whereas DNSSEC is primarily designed to protect consumers of DNS response messages, this protocol is designed to protect consumers of zones.

There are existing tools and protocols that provide data security, such as OpenPGP [RFC4880] and S/MIME [RFC3851]. In fact, the internic.net site publishes PGP signatures along side the root zone and other files available there. However, this is a detached signature with no strong association to the corresponding zone file other than its timestamp. Non-detached signatures are, of course, possible, but these necessarily change the format of the file being distributed. That is, a zone signed with OpenPGP or S/MIME no longer looks like a DNS zone and could not directly be loaded into a name server. Once loaded the signature data is lost, so it does not survive further propagation.

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

1.2. Design Overview

This document introduces a new Resource Record type designed to convey a message digest of the content of a zone. The digest is calculated at the time of zone publication. Ideally the zone is signed with DNSSEC to guarantee that any modifications of the digest can be detected. The procedures for digest calculation and DNSSEC

signing are similar (i.e., both require the same ordering of RRs) and can be done in parallel.

The zone digest is designed to be used on zones that are relatively stable and have infrequent updates. As currently specified, the digest is re-calculated over the entire zone content each time. This specification does not provide an efficient mechanism for incremental updates of zone data. It does, however, reserve a field in the ZONEMD record for future work to support incremental zone digest algorithms (e.g. using Merkle trees).

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

1.3. Use Cases

1.3.1. Root Zone

The root zone [InterNIC] is one of the most widely distributed DNS zone on the Internet, served by 930 separate instances [RootServers] at the time of this writing. Additionally, many organizations configure their own name servers to serve the root zone locally. Reasons for doing so include privacy and reduced access time. [RFC7706] describes one, but not the only, way to do this. As the root zone spreads beyond its traditional deployment boundaries, the need for verification of the completeness of the zone contents becomes increasingly important.

1.3.2. Providers, Secondaries, and Anycast

Since its very early days, the developers of the DNS recognized the importance of secondary name servers and service diversity. However, they may not have anticipated the complexity of modern DNS service provisioning which can include multiple third-party providers and hundreds of anycast instances. Instead of a simple primary-to-secondary zone distribution system, today it is possible to have multiple levels, multiple parties, and multiple protocols involved in the distribution of zone data. This complexity introduces new places for problems to arise. The zone digest protects the integrity of data that flows through such systems.

1.3.3. Response Policy Zones

DNS Response Policy Zones is "a method of expressing DNS response policy information inside specially constructed DNS zones..." [RPZ]. A number of companies provide RPZ feeds, which can be consumed by name server and firewall products. Since these are zones, AXFR is often, but not necessarily used for transmission. While RPZ zones can certainly be signed with DNSSEC, the data is not queried directly, and would not be subject to DNSSEC validation.

1.3.4. Centralized Zone Data Service

ICANN operates the Centralized Zone Data Service [CZDS], which is a repository of top-level domain zone files. Users request access to the system, and to individual zones, and are then able to download zone data for certain uses. Adding a zone digest to these would provide CZDS users with assurances that the data has not been modified. Note that ZONEMD could be added to CZDS zone data independently of the zone served by production name servers.

1.3.5. General Purpose Comparison Check

Since the zone digest does not depend on presentation format, it could be used to compare multiple copies of a zone received from different sources, or copies generated by different processes.

1.4. Requirements Language

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

2. The ZONEMD Resource Record

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

FOR DISCUSSION: This document is currently written as though a zone MUST NOT contain more than one ZONEMD RR. Having exactly one ZONEMD record per zone simplifies this protocol and eliminates confusion around downgrade attacks, at the expense of algorithm agility.

2.2. ZONEMD Presentation Format

The presentation format of the RDATA portion is as follows:

The Serial field **MUST** be represented as an unsigned decimal integer.

The Digest Type field **MUST** be represented as an unsigned decimal integer.

The Reserved field **MUST** be represented as an unsigned decimal integer set to zero.

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

2.3. ZONEMD Example

The following example shows a ZONEMD RR.

```
example.com. 86400 IN ZONEMD 2018031500 4 0 (
    FEBE3D4CE2EC2FFA4BA99D46CD69D6D29711E55217057BEE
    7EB1A7B641A47BA7FED2DD5B97AE499FAFA4F22C6BD647DE )
```

3. Calculating the Digest

3.1. Canonical Format and Ordering

Calculation of the zone digest **REQUIRES** the RRs in a zone to be processed in a consistent format and ordering. Correct ordering of the zone depends on (1) ordering of owner names in the zone, (2) ordering of RRsets with the same owner name, and (3) ordering of RRs within an RRset.

This specification adopts DNSSEC's canonical ordering for names (Section 6.1 of [RFC4034]), and canonical ordering for RRs within an RRset (Section 6.3 of [RFC4034]). It also adopts DNSSEC's canonical RR form (Section 6.2 of [RFC4034]). However, since DNSSEC does not define a canonical ordering for RRsets having the same owner name, that ordering is defined here.

3.1.1. Order of RRsets Having the Same Owner Name

For the purposes of calculating the zone digest, RRsets having the same owner name **MUST** be numerically ordered, in ascending order, by their numeric RR TYPE.

3.1.2. Duplicate RRs

As stated in Section 5 of [RFC2181], it is meaningless for a zone to have multiple RRs with equal owner name, class, type, and RDATA. In the interest of consistency and interoperability, such duplicate RRs MUST NOT be included in the calculation of a zone digest.

3.2. Add ZONEMD Placeholder

In preparation for calculating the zone digest, any existing ZONEMD record at the zone apex MUST first be deleted.

FOR DISCUSSION: Should non-apex ZONEMD records be allowed in a zone? Or forbidden?

Prior to calculation of the digest, and prior to signing with DNSSEC, a placeholder ZONEMD record MUST be added to the zone apex. This serves two purposes: (1) it allows the digest to cover the Serial, Digest Type, and Reserved field values, and (2) ensures that appropriate denial-of-existence (NSEC, NSEC3) records are created if the zone is signed with DNSSEC.

It is RECOMMENDED that the TTL of the ZONEMD record match the TTL of the SOA.

In the placeholder record, the Serial field MUST be set to the current SOA Serial. The Digest Type field MUST be set to the value for the chosen digest algorithm. The Reserved field MUST be set to zero. The Digest field MUST be set to all zeroes and of length appropriate for the chosen digest algorithm.

3.3. Optionally Sign the Zone

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

3.4. Calculate the Digest

The zone digest is calculated by concatenating the canonical on-the-wire form (without name compression) of all RRs in the zone, in the order described above, subject to the inclusion/exclusion rules described below, and then applying the digest algorithm:

digest = digest_algorithm(RR(1) | RR(2) | RR(3) | ...)

where "|" denotes concatenation, and

RR(i) = owner | type | class | TTL | RDATA length | RDATA

3.4.1. Inclusion/Exclusion Rules

When calculating the digest, the following inclusion/exclusion rules apply:

- o All records in the zone, including glue records, MUST be included.
- o Occluded data ([RFC5936] Section 3.5) MUST be included.
- o Duplicate RRs with equal owner, class, type, and RDATA MUST NOT be included.
- o The placeholder ZONEMD RR MUST be included.
- o If the zone is signed, DNSSEC RRs MUST be included, except:
- o The RRSIG covering ZONEMD MUST NOT be included.

3.5. Update ZONEMD RR

Once the zone digest has been calculated, its value is then copied to the Digest field of the ZONEMD record.

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

Some implementations of incremental DNSSEC signing might update the zone's serial number for each resigning. However, to preserve the calculated digest, generation of the ZONEMD signature at this time MUST NOT also result in a change of the SOA serial number.

4. Verifying Zone Message Digest

The recipient of a zone that has a message digest record can verify the zone by calculating the digest as follows:

1. The verifier SHOULD first determine whether or not to expect DNSSEC records in the zone. This can be done by examining locally configured trust anchors, or querying for (and

validating) DS RRs in the parent zone. For zones that are provably unsigned, digest validation continues at step 4 below.

2. For zones that are provably signed, the existence of the apex ZONEMD record MUST be verified. If the ZONEMD record provably does not exist, digest verification cannot be done. If the ZONEMD record does provably exist, but is not found in the zone, digest verification MUST NOT be considered successful.
3. For zones that are provably signed, the SOA RR and ZONEMD RR MUST have valid signatures, chaining up to a trust anchor. If DNSSEC validation of the SOA or ZONEMD records fails, digest verification MUST NOT be considered successful.
4. If the zone contains more than one apex ZONEMD RR, digest verification MUST NOT be considered successful.
5. The SOA Serial field MUST exactly match the ZONEMD Serial field. If the fields do not match, digest verification MUST NOT be considered successful.
6. The ZONEMD Digest Type field MUST be checked. If the verifier does not support the given digest type, it SHOULD report that the zone digest could not be verified due to an unsupported algorithm.
7. The Reserved field MUST be checked. If the Reserved field value is not zero, verification MUST NOT be considered successful.
8. The received Digest Type and Digest values are copied to a temporary location.
9. The ZONEMD RR's RDATA is reset to the placeholder values described in Section 3.2.
10. The zone digest is computed over the zone data as described in Section 3.4.
11. The calculated digest is compared to the received digest stored in the temporary location. If the two digest values match, verification is considered successful. Otherwise, verification MUST NOT be considered successful.
12. The ZONEMD RR's RDATA is reset to the received Digest Type and Digest stored in the temporary location. Thus, any downstream clients can similarly verify the zone.

5. Scope of Experimentation

This memo is published as an Experimental RFC. The purpose of the experimental period is to provide the community time to analyze and evaluate to the methods defined in this document, particularly with regard to the wide variety of DNS zones in use on the Internet.

Additionally, the ZONEMD record defined in this document includes a Reserved field in the form of an 8-bit integer. The authors have a particular future use in mind for this field, namely to support efficient digests in large, dynamic zones. We intend to conduct future experiments using Merkle trees of varying depth. The choice of tree depth can be encoded in this reserved field. We expect values for tree depth to range from 0 to 10, requiring at most four bits of this field. This leaves another four bits available for other future uses, if absolutely necessary.

FOR DISCUSSION: The authors are willing to remove the Reserved field from this specification if the working group would prefer it. It would mean, however, that a future version of this protocol designed to efficiently support large, dynamic zones would most likely require a new RR type.

The duration of the experiment is expected to be no less than two years from the publication of this document. If the experiment is successful, it is expected that the findings of the experiment will result in an updated document for Standards Track approval.

6. IANA Considerations

6.1. ZONEMD RRtype

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

Type: ZONEMD

Value: 63

Meaning: Message Digest Over Zone Data

Reference: This document

6.2. ZONEMD Digest Type

This document asks IANA to create a new "ZONEMD Digest Types" registry with initial contents as follows:

Value	Description	Status	Reference
1	SHA384	Mandatory	[RFC6605]

Table 1: ZONEMD Digest Types

7. Security Considerations

7.1. Attacks Against the Zone Digest

The zone digest allows the receiver to verify that the zone contents haven't been modified since the zone was generated/published. Verification is strongest when the zone is also signed with DNSSEC. An attacker, whose goal is to modify zone content before it is used by the victim, may consider a number of different approaches.

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

The attacker might perform a downgrade attack by removing the ZONEMD record. This is why Section 4 REQUIRES that the verifier checks DNSSEC denial-of-existence proofs in step 2.

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

7.2. Attacks Utilizing the Zone Digest

Nothing in this specification prevents clients from making, and servers from responding to, ZONEMD queries. One might consider how well ZONEMD responses could be used in a distributed denial-of-service amplification attack.

The ZONEMD RR is moderately sized, much like the DS RR. A single ZONEMD RR contributes approximately 40 to 65 octets to a DNS response, for currently defined digest types. Certainly other query types result in larger amplification effects (i.e., DNSKEY).

8. Privacy Considerations

This specification has no impacts on user privacy.

9. Acknowledgments

The authors wish to thank David Blacka, Scott Hollenbeck, and Rick Wilhelm for providing feedback on early drafts of this document. Additionally, they thank Joe Abley, Mark Andrews, Olafur Gudmundsson, Paul Hoffman, Evan Hunt, Shumon Huque, Tatuya Jinmei, Burt Kaliski, Shane Kerr, Matt Larson, John Levine, Ed Lewis, Mukund Sivaraman, Petr Spacek, Ondrej Sury, Florian Weimer, Tim Wicinski, Paul Wouters, and other members of the dnsop working group for their input.

10. Implementation Status

10.1. Authors' Implementation

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

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

This implementation does not:

- o Perform DNSSEC validation of the ZONEMD record.
- o Support the Gost digest algorithm.
- o Output the ZONEMD record in its defined presentation format.

10.2. Shane Kerr's Implementation

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

- o Read an input zone and a output zone with ZONEMD record.
- o Verify the zone digest from an input zone.

- o Output the ZONEMD record in its defined presentation format.
- o Generate Gost digests.

This implementation does not:

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

11. Change Log

RFC Editor: Please remove this section.

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

From -00 to -01:

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

From -01 to -02:

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

From -02 to -03:

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

From -03 to -04:

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

From -04 to -05:

- o Made SHA384 the only supported ZONEMD digest type.
- o Disassociated ZONEMD digest types from DS digest types.
- o Updates to Introduction based on list feedback.
- o Changed "zone file" to "zone" everywhere.
- o Restored text about why ZONEMD has a Serial field.
- o Clarified ordering of RRsets having same owner to be numerically ascending.
- o Clarified that all duplicate RRs (not just SOA) must be suppressed in digest calculation.

- o Clarified that the Reserved field must be set to zero and checked for zero in verification.
- o Clarified that occluded data must be included.
- o Clarified procedure for verification, using temporary location for received digest.
- o Explained why Reserved field is 8-bits.
- o IANA Considerations section now more specific.
- o Added complex zone to examples.
- o

From -05 to -06:

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

12. References

12.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <<https://www.rfc-editor.org/info/rfc2181>>.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<https://www.rfc-editor.org/info/rfc4034>>.

- [RFC6605] Hoffman, P. and W. Wijngaards, "Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC", RFC 6605, DOI 10.17487/RFC6605, April 2012, <<https://www.rfc-editor.org/info/rfc6605>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

12.2. Informative References

- [CZDS] Internet Corporation for Assigned Names and Numbers, "Centralized Zone Data Service", October 2018, <<https://czds.icann.org/>>.
- [dns-over-https] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", draft-ietf-doh-dns-over-https-12 (work in progress), June 2018, <<https://tools.ietf.org/html/draft-ietf-doh-dns-over-https-12>>.
- [InterNIC] ICANN, "InterNIC FTP site", May 2018, <<ftp://ftp.internic.net/domain/>>.
- [ldns-zone-digest] Verisign, "Implementation of Message Digests for DNS Zones using the ldns library", July 2018, <<https://github.com/verisign/ldns-zone-digest>>.
- [RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, DOI 10.17487/RFC1995, August 1996, <<https://www.rfc-editor.org/info/rfc1995>>.
- [RFC2065] Eastlake 3rd, D. and C. Kaufman, "Domain Name System Security Extensions", RFC 2065, DOI 10.17487/RFC2065, January 1997, <<https://www.rfc-editor.org/info/rfc2065>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<https://www.rfc-editor.org/info/rfc2136>>.
- [RFC2535] Eastlake 3rd, D., "Domain Name System Security Extensions", RFC 2535, DOI 10.17487/RFC2535, March 1999, <<https://www.rfc-editor.org/info/rfc2535>>.

- [RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <<https://www.rfc-editor.org/info/rfc2845>>.
- [RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <<https://www.rfc-editor.org/info/rfc2931>>.
- [RFC3851] Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, DOI 10.17487/RFC3851, July 2004, <<https://www.rfc-editor.org/info/rfc3851>>.
- [RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <<https://www.rfc-editor.org/info/rfc4880>>.
- [RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <<https://www.rfc-editor.org/info/rfc5936>>.
- [RFC7706] Kumari, W. and P. Hoffman, "Decreasing Access Time to Root Servers by Running One on Loopback", RFC 7706, DOI 10.17487/RFC7706, November 2015, <<https://www.rfc-editor.org/info/rfc7706>>.
- [RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <<https://www.rfc-editor.org/info/rfc7719>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RootServers]
Root Server Operators, "Root Server Technical Operations", July 2018, <<https://www.root-servers.org/>>.
- [RPZ] Vixie, P. and V. Schryver, "DNS Response Policy Zones (RPZ)", draft-vixie-dnsop-dns-rpz-00 (work in progress), June 2018, <<https://tools.ietf.org/html/draft-vixie-dnsop-dns-rpz-00>>.

[ZoneDigestHackathon]

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

Appendix A. Example Zones With Digests

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

A.1. Simple EXAMPLE Zone

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

```
example.      86400    IN    SOA      ns1 admin 2018031900 (
                                1800 900 604800 86400 )
              86400    IN    NS       ns1
              86400    IN    NS       ns2
              86400    IN    ZONEMD    2018031900 1 0 (
                                f32765ce15c50477
                                42a08be15d9a0efb
                                749417eaadcfa28b
                                1bf751b6bc49f9be
                                a615c4a386cfd6a5
                                d85e2d2182691249 )
ns1           3600     IN    A        127.0.0.1
ns2           3600     IN    AAAA     ::1
```

A.2. Complex EXAMPLE Zone

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

```

example.      86400    IN    SOA      ns1 admin 2018031900 (
                1800 900 604800 86400 )
                86400    IN    NS       ns1
                86400    IN    NS       ns2
                86400    IN    ZONEMD   2018031900 1 0 (
                686a6d74d5638612
                64ea4e6cc12c22d1
                7ebc529791d393bd
                e164a45390f714e9
                9ede0d05a5644573
                da4bbcc83744acf2 )
ns1           3600     IN    A        127.0.0.1
ns2           3600     IN    AAAA     ::1
occluded.sub  7200     IN    TXT     "I'm occluded but must be digested"
sub           7200     IN    NS       ns1
duplicate     300      IN    TXT     "I must be digested just once"
duplicate     300      IN    TXT     "I must be digested just once"
foo.test.     555      IN    TXT     "out-of-zone data must be excluded"

```

A.3. The URI.ARPA Zone

The URI.ARPA zone retrieved 2018-10-21.

```

; <<>> DiG 9.9.4 <<>> @lax.xfr.dns.icann.org uri.arpa axfr
; (2 servers found)
;; global options: +cmd
uri.arpa.      3600     IN    SOA      sns.dns.icann.org. (
                noc.dns.icann.org. 2018100702 10800 3600 1209600 3600 )
uri.arpa.      3600     IN    RRSIG     NSEC 8 2 3600 (
                20181028142623 20181007205525 47155 uri.arpa.
                eEC4w/oXLRlEpwgv4MBiDtSBsXhqrJVvJWUpbX8XpetAvD35bxwNCUTi
                /pAJVUXefegWeiriD2rkTgCBCMmn7YQIm3gdR+HjY/+o3BXNQnz97f+e
                HAE9EDDzoNVfL1PyV/2fde9tDeUuAGVVwmD399NGq9jWYMRpyri2kysr q/g= )
uri.arpa.      86400    IN    RRSIG     NS 8 2 86400 (
                20181028172020 20181007175821 47155 uri.arpa.
                ATyV2A2A8ZoggC+68u4GuP5MOUuR+2rr3eWOkEU55zAHld/7FiBxl4ln
                4byJYy7NudUwlMOEXajqFZE7DVl8PpcvrP3HeeGaVzKqaWj+aus0jbKF
                Bsvs2blqDZemBfkz/IfAhUTJKnto0vSUicJKfItu0GjyYNJCz2CqEuGD Wxc= )
uri.arpa.      600      IN    RRSIG     MX 8 2 600 (
                20181028170556 20181007175821 47155 uri.arpa.
                e7/r3KXDohX1lyVavetFFObp8fB8aXT76HnN9KCQDxSnSghNM83UQV0t
                1TtD8JVeNlmCvcNFZpagwIgB7XhTtm6Beur/m5ES+4uSnVeS6Q66HBZK
                A3mR95IpevuVIZvvJ+GcCAQpBo6KRODYvJ/c/ZG6sfYWkZ7qg/Em5/+3 4UI= )
uri.arpa.      3600     IN    RRSIG     DNSKEY 8 2 3600 (
                20181028152832 20181007175821 15796 uri.arpa.
                nzpbnh0OqsgBBP8St28pLvPEQ3wZAUdEBuUwil+rtjjWlYYiqjPxZ286
                XF4Rqlusfv5x71jZz5IqswOaQgia91ylodFpLuXD6FTGs2nXGhNKkg1V

```

```

chHgtwj70mXU72GefVgo8TxrFYzxuEFP5ZTP92t97FVWVYyF86sbbR
6DZj3uA2wEvqBVLECGJLrMQ9Yy7MueJl3UA4h4E6zO2JY9Yp0W9woq0B
dqkkwYTwzogyYffPmGAJG91RJ2h6cHtFjEZe2MnaY2glqniZ0WT9vXXd
uFPm0KD9U77Ac+ZtctAF9tsZwSdAoL365E2L1usZbA+K0BnPPqGFJRJk
5R0A1w== )
uri.arpa. 3600 IN RRSIG DNSKEY 8 2 3600 (
20181028152832 20181007175821 55480 uri.arpa.
1WtQV/5szQjkXmbcD47/+rOW8kJPksRFHlzxmxzt906+DBYyfrH6uq5X
nHvrUlQO6M12uhqDeL+bDFVgqSpNy+42/OaZvaK3J8EzPZVBHPJykKmv
63T83aAiJrAyHzOaEdmzLCpalqcEE2ImzlLHSafManRfJL8Yuv+JDZFj
2WDWfEcUuwkmIzWX11zxp+DxwzyUlRl7x4+ok5iKZWig5UnBAf6B8T75
WnXzlhCw3F2pXI0a5LYg71L3Tp/xhjN6Yy9jG1IRf5BjB59X2zra3a2R
PkI09SSnuEwHyFlmDaV5BmQrLGRnCjvwXA7ho2m+vv4SP5dUdXf+GTaA
1HeBfw== )
uri.arpa. 3600 IN RRSIG SOA 8 2 3600 (
20181029114753 20181008222815 47155 uri.arpa.
qn8yBNoHDjGdT79U2Wu9IIahoS0YPOgYP8lG+qwPcrZ1BwGiHywuoUa2
Mx6BWZlg+HDyaxj2iOmx+IIqoUhhXUBo7IUkJFlgrOKCgAR2twDHRXu
9BUQHy9SoVl6wYm3kBTEPyxW5FFm8vcdnKAF7sxSY8BbaYNpRIEjDx4A JUC= )
uri.arpa. 3600 IN NSEC ftp.uri.arpa. NS SOA (
MX RRSIG NSEC DNSKEY )
uri.arpa. 86400 IN NS a.iana-servers.net.
uri.arpa. 86400 IN NS b.iana-servers.net.
uri.arpa. 86400 IN NS c.iana-servers.net.
uri.arpa. 86400 IN NS ns2.lacnic.net.
uri.arpa. 86400 IN NS sec3.apnic.net.
uri.arpa. 600 IN MX 10 pechora.icann.org.
uri.arpa. 3600 IN DNSKEY 256 3 8 (
AwEAAcBi7tSart2J599zbYWspMNGN70IBWb4ziqyQYH9MTB/VCz6WyUK
uXunwiJJBbQ3bcLqTLWEw134B6cTMHrZpjTab5WAwg4XcWUu8mdcPTiL
Bl6qVRlRD0WiFCTzuYUfkwshlRbr7rvrxSQhF5rh71zSpwV5jjjp65Wx
SdJjlH0B )
uri.arpa. 3600 IN DNSKEY 257 3 8 (
AwEAAbNVv6ulgRdO31MtAehz7j3ALRjwZglWesnzvllQl/+hBRZr9QoY
cO2I+DkO4Q1NKxox4DUIxj8SxPO3GwDuOFR9q2/CFi2O0mZjafbdYtWc
3zSdBbi3q0cwCIx7GuG9eqLL+pg7mdk9dgdNZfHwB0LnqTD8ebLpSrO/
Id7kBaiqYOfMlZnh2fp+2h6OOJZHtY0DK1UlssyB5PKsE0tVzo5s6zo9
iXKe5u+8WTMaGDY49vG80JPAKE7ezMiH/NZcUMiE0PRZ8D3foq2dYuS5
ym+vA83Z7v8A+Rwh4UGnjxKB8zmr803V0ASAmHz/gwH5Vb0nH+LObwFt
l3wpbp+Wpm8= )
uri.arpa. 3600 IN DNSKEY 257 3 8 (
AwEAAbwnFTakCvaUKsXji4mgmxZUJi1IygbnGahbkmFEa0L16J+TchKR
wcgzVfsxUGa2MmeA4hgkAooC3uy+tTmoMsgy8uq/JA24DjiHzd46LfD
FK/qMidVqFpYSheq2Vv5ojkuIsx4oe4KsafGWYN0czKZgH5loGjN2aJG
mrIm++XCphOskGcsQYl65MIzuXffzJyxlAut+ecAIiVeqRaqQfr8LRU
7wIsLxinXirprtQrbor+EtvLHp9qXE6ARTZDzf4jvsNpKvLFZtmxzFf3
e/UJz5eHjpwDSiZL7xE8aE1o1nGfPtJx9ZnB3bapltaj5wY+5XOCKgY0
xmJVvNQlwdE= )

```

```

ftp.uri.arpa.      3600      IN          RRSIG      NSEC 8 3 3600 (
  20181028080856 20181007175821 47155 uri.arpa.
  HClGAqPxzkYkAT7Q/QNtQeB6YrkP6EPOef+9Qo5/2zngwAewXEAQiyF9
  jD1USJirom11QqBS3v3aIdW/LXORs4Ez3hLcKNO1cKHsOuWAqzmE+BPP
  Arfh8N95jqh/q6vpaB9UtMkQ53tM2fYU1GszOLN0knxbHgDHAh2axMGH 1qM= )
ftp.uri.arpa.      604800   IN          RRSIG      NAPTR 8 3 604800 (
  20181028103644 20181007205525 47155 uri.arpa.
  WoLi+vZzkxaoLr2IGZnwkRvcDf6KxiWQd1WZP/U+AWnV+7MiqsWPZaf0
  9toRErerGoFOiOASNxZjBGJrRgjmavOM9U+LZSconP9zrNFd4dIu6kp5
  YxlQJ0uHOvx1ZHFCj6lAt1ACUIw04ZhMydTmi27c8MzEOMepvn7iH7r7 k7k= )
ftp.uri.arpa.      3600      IN          NSEC       http.uri.arpa. NAPTR (
  RRSIG NSEC )
ftp.uri.arpa.      604800   IN          NAPTR      0 0 "" "" (
  "!^ftp://([^:/?#]*).*$!\\1!i" . )
http.uri.arpa.     3600      IN          RRSIG      NSEC 8 3 3600 (
  20181029010647 20181007175821 47155 uri.arpa.
  U03NntQ73LHWpfLmUK8nMsqkwVsOGW2KdsyuHYAjqQSZvKbtmbv7HBmE
  H1+Ii3Z+wtfdMZBy5aC/6sHdx69BfZJs16xumycMlAy6325DKTQbIMN+
  ift9GrKBC7cgCd2msF/uzSrYxxg4MJQzBPv1kwXnY3b7eJS1IXisBIn7 3b8= )
http.uri.arpa.     604800   IN          RRSIG      NAPTR 8 3 604800 (
  20181029011815 20181007205525 47155 uri.arpa.
  T7mRrdag+WSmG+n22mtBSQ/0Y3v+rdDnfQV90LN5Fq32N5K2iYFaj7F7
  Tp56oOznytfcL4fHrqOE0wRc9NWOCUec9C7WalqJQc11EvgoAM+L6f0
  RsEjWq6+9jv1LKMXQv0xQuMX17338uoD/xiAFQSnDbiQKxwWMqVAimv5 7Zs= )
http.uri.arpa.     3600      IN          NSEC       mailto.uri.arpa. NAPTR (
  RRSIG NSEC )
http.uri.arpa.     604800   IN          NAPTR      0 0 "" "" (
  "!^http://([^:/?#]*).*$!\\1!i" . )
mailto.uri.arpa.   3600      IN          RRSIG      NSEC 8 3 3600 (
  20181028110727 20181007175821 47155 uri.arpa.
  GvxzVL85rEukwGqtuLxek9ipwjBMfTOFIEyJ7afC8HxVMs6mfFa/nEM/
  IdFvvFg+lcYoJSQYusAVYF13xPbgrxVSLK125QutCFMdc/YjuZENq5cl
  fQciMRD7R3+znZfm8d8u/snLV9w4D+1TBZrJJUBelEfc8vum5vvV7819 ZoY= )
mailto.uri.arpa.   604800   IN          RRSIG      NAPTR 8 3 604800 (
  20181028141825 20181007205525 47155 uri.arpa.
  MaADUgc3fc5v++M0YmqjGk3jBdfIA5RuP62hUS1PsFZO4k37erjIGCfF
  j+g84yc+QgbSde0PQHszl9fE/+SU5ZXiS9YdcbzSZxp2erFpZOTchrpg
  916T4vx6i59scodjb016bDyZ+mtIPrc1w6b4hUyOUTsDQoAJYxdfEuMg Vy4= )
mailto.uri.arpa.   3600      IN          NSEC       urn.uri.arpa. NAPTR (
  RRSIG NSEC )
mailto.uri.arpa.   604800   IN          NAPTR      0 0 "" "" (
  "!^mailto:(.*)@(.*)$!\\2!i" . )
urn.uri.arpa.      3600      IN          RRSIG      NSEC 8 3 3600 (
  20181028123243 20181007175821 47155 uri.arpa.
  Hgsw4Deops108uWyELGe6hpR/OEqCnTHvahlwiQkHhO5CSEQrbhmFAWe
  UOkmGAdTEYrSz+skLRQuITRMwzyFf4oUkZihGyhZyzHbcxWfuDc/Pd/9
  DS156gdeBwylevn5wBTms8yWQVknTphbJH395gRqZuaJs3LD/qTyJ5Dp LvA= )
urn.uri.arpa.      604800   IN          RRSIG      NAPTR 8 3 604800 (

```

```

20181029071816 20181007205525 47155 uri.arpa.
ALIZD0vBqAQQt40GQ0Efaj8OCyE9xSRJRdyvyn/H/wZVXFRFKrQYrLAS
D/K7q6CMT0xTRCu2J8yes63WJiaJEdnh+dscXzZkmOg4n5PsgZbkvUSW
BiGtxvz5jNncM0xVbkjbtByrvJQAO1cU1mnlDKelFmVB1uLpVdA9Ib4J hMU= )
urn.uri.arpa.      3600      IN          NSEC      uri.arpa. NAPTR RRSIG (
NSEC )
urn.uri.arpa.      604800   IN          NAPTR      0 0 "" "" (
"/urn:([^:]+)/\\1/i" . )
uri.arpa.          3600      IN          SOA          sns.dns.icann.org. (
noc.dns.icann.org. 2018100702 10800 3600 1209600 3600 )
;; Query time: 66 msec
;; SERVER: 192.0.32.132#53(192.0.32.132)
;; WHEN: Sun Oct 21 20:39:28 UTC 2018
;; XFR size: 34 records (messages 1, bytes 3941)
uri.arpa.          3600      IN          ZONEMD      2018100702 1 0 (
80af7afd9540ff2c4c559f0d2b83393386304e093e0e66787378b2
a578297b49b4dcc422bce2c300bb92b354575283a )

```

A.4. The ROOT-SERVERS.NET Zone

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

```
root-servers.net.      3600000 IN  SOA      a.root-servers.net. (
    nstld.verisign-grs.com. 2018091100 14400 7200 1209600 3600000 )
root-servers.net.      3600000 IN  NS       a.root-servers.net.
root-servers.net.      3600000 IN  NS       b.root-servers.net.
root-servers.net.      3600000 IN  NS       c.root-servers.net.
root-servers.net.      3600000 IN  NS       d.root-servers.net.
root-servers.net.      3600000 IN  NS       e.root-servers.net.
root-servers.net.      3600000 IN  NS       f.root-servers.net.
root-servers.net.      3600000 IN  NS       g.root-servers.net.
root-servers.net.      3600000 IN  NS       h.root-servers.net.
root-servers.net.      3600000 IN  NS       i.root-servers.net.
root-servers.net.      3600000 IN  NS       j.root-servers.net.
root-servers.net.      3600000 IN  NS       k.root-servers.net.
root-servers.net.      3600000 IN  NS       l.root-servers.net.
root-servers.net.      3600000 IN  NS       m.root-servers.net.
a.root-servers.net.    3600000 IN  AAAA     2001:503:ba3e::2:30
a.root-servers.net.    3600000 IN  A        198.41.0.4
b.root-servers.net.    3600000 IN  MX       20 mail.isi.edu.
b.root-servers.net.    3600000 IN  AAAA     2001:500:200::b
b.root-servers.net.    3600000 IN  A        199.9.14.201
c.root-servers.net.    3600000 IN  AAAA     2001:500:2::c
c.root-servers.net.    3600000 IN  A        192.33.4.12
d.root-servers.net.    3600000 IN  AAAA     2001:500:2d::d
d.root-servers.net.    3600000 IN  A        199.7.91.13
e.root-servers.net.    3600000 IN  AAAA     2001:500:a8::e
e.root-servers.net.    3600000 IN  A        192.203.230.10
f.root-servers.net.    3600000 IN  AAAA     2001:500:2f::f
f.root-servers.net.    3600000 IN  A        192.5.5.241
g.root-servers.net.    3600000 IN  AAAA     2001:500:12::d0d
g.root-servers.net.    3600000 IN  A        192.112.36.4
h.root-servers.net.    3600000 IN  AAAA     2001:500:1::53
h.root-servers.net.    3600000 IN  A        198.97.190.53
i.root-servers.net.    3600000 IN  MX       10 mx.i.root-servers.org.
i.root-servers.net.    3600000 IN  AAAA     2001:7fe::53
i.root-servers.net.    3600000 IN  A        192.36.148.17
j.root-servers.net.    3600000 IN  AAAA     2001:503:c27::2:30
j.root-servers.net.    3600000 IN  A        192.58.128.30
k.root-servers.net.    3600000 IN  AAAA     2001:7fd::1
k.root-servers.net.    3600000 IN  A        193.0.14.129
l.root-servers.net.    3600000 IN  AAAA     2001:500:9f::42
l.root-servers.net.    3600000 IN  A        199.7.83.42
m.root-servers.net.    3600000 IN  AAAA     2001:dc3::35
m.root-servers.net.    3600000 IN  A        202.12.27.33
root-servers.net.      3600000 IN  SOA      a.root-servers.net. (
    nstld.verisign-grs.com. 2018091100 14400 7200 1209600 3600000 )
root-servers.net.      3600000 IN  ZONEMD   2018091100 1 0 (
    aadf7a017bccd8cefe6040494800249fd5edc3d49e2e8ce8db7522f47f
    97f168db794bf5f679fbe0c8433fb66f7a0c26 )
```

Authors' Addresses

Duane Wessels
Verisign
12061 Bluemont Way
Reston, VA 20190

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

Piet Barber
Verisign
12061 Bluemont Way
Reston, VA 20190

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

Matt Weinberg
Verisign
12061 Bluemont Way
Reston, VA 20190

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

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043

Email: warren@kumari.net

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

Email: ietf@hardakers.net

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

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

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

Abstract

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

Ed note

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 22, 2020.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

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

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

1. Introduction

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

For example, consider the following record:

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

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

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

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

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

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

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

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

1.1. Background and Terminology

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

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

2. The BULK Resource Record

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

The Type value for the BULK RR type is TBD.

The BULK RR is class-independent.

2.1. BULK RDATA Wire Format

The RDATA for a BULK RR is as follows:

```

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

```

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

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

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

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

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

string         = 1*(ctext / quoted-char)

decnum         = 1*decdigit
                ; constrained to 65535 maximum.

hexnum         = 1*hexdigit
                ; constrained to ffff maximum.

octet          = %x00-FF

decdigit       = %x30-39
                ; 0-9

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

ctext          = <any octet excepting "\">

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

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

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

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

```

replace      = 1*(reference / string)

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

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

posrange     = position "-" position

position     = 1*decnum

options      = delimiter [interval [padding]]

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

interval     = "|" *2decdigit

padding      = "|" *2decdigit

```

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

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

2.2. The BULK RR Presentation Format

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

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

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

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

3. BULK Replacement

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

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

3.1. Matching the Domain Name Pattern

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

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

3.2. Record Generation using Replacement Pattern

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

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

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

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

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

3.2.1. Delimiters

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

3.2.2. Delimiter intervals

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

3.2.3. Padding length

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

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

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

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

3.2.4. Final processing

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

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

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

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

4. Known Limitations

This section defines known limitations of the BULK resource type.

4.1. Unsupported Nameservers

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

5. Security Considerations

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

5.1. DNSSEC Signature Strategies

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

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

5.1.1. On-the-fly Signatures

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

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

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

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

5.1.2. Alternative Signature Scheme

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

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

5.1.3. Non-DNSSEC Zone Support Only

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

5.2. DDOS Attack Vectors and Mitigation

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

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

5.3. Implications of Large-Scale DNS Records

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

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

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

6. Privacy Considerations

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

7. IANA Considerations

IANA is requested to assign numbers for the BULK RR.

8. Acknowledgments

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

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

9. References

9.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <<https://www.rfc-editor.org/info/rfc2181>>.

- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <<https://www.rfc-editor.org/info/rfc2308>>.
- [RFC2317] Eidnes, H., de Groot, G., and P. Vixie, "Classless IN-ADDR.ARPA delegation", BCP 20, RFC 2317, DOI 10.17487/RFC2317, March 1998, <<https://www.rfc-editor.org/info/rfc2317>>.
- [RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <<https://www.rfc-editor.org/info/rfc3597>>.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.

9.2. Informative References

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

[RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <<https://www.rfc-editor.org/info/rfc7719>>.

Appendix A. BULK Examples

A.1. Example 1

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

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

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

A.2. Example 2

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

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

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

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

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

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

[Change this to a hex matching example?]

A.3. Example 3

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

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

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

A.4. Example 4

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

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

A.5. Example 5

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

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

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

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

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

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

Authors' Addresses

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

Email: John.Woodworth@CenturyLink.com

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

Email: Dean.Ballew@CenturyLink.com

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

Email: shashwath.bindinganaveliraghavan@hughes.com

David C Lawrence
Oracle

Email: tale@dd.org