Workgroup: Internet Engineering Task Force
Internet-Draft:
draft-ietf-dnsop-caching-resolution-
failures-01
Updates: 2308 (if approved)
Published: 12 September 2022
Intended Status: Standards Track
Expires: 16 March 2023
Authors: D. Wessels W. Carroll M. Thomas
Verisign Verisign Verisign

Negative Caching of DNS Resolution Failures

Abstract

In the DNS, resolvers employ caching to reduce both latency for end users and load on authoritative name servers. The process of resolution may result in one of three types of responses: (1) a response containing the requested data; (2) a response indicating the requested data does not exist; or (3) a non-response due to a resolution failure in which the resolver does not receive any useful information regarding the data's existence. This document concerns itself only with the third type.

RFC 2308 specifies requirements for DNS negative caching. There, caching of type (1) and (2) responses is mandatory and caching of type (3) responses is optional. This document updates RFC 2308 to require negative caching for DNS resolution failures.

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 16 March 2023.

Table of Contents

1. Introduction

Caching has always been a fundamental component of DNS resolution on the Internet. For example [RFC0882] states:

"The sheer size of the database and frequency of updates suggest that it must be maintained in a distributed manner, with local caching to improve performance."

The early DNS RFCs ([RFC0882], [RFC0883], [RFC1034], and [RFC1035]) primarily discuss caching in the context of what [RFC2308] calls "positive" responses, that is, when the response includes the

requested data. In this case, a TTL is associated with each resource record in the response. Resolvers can cache and reuse the data until the TTL expires.

Section 4.3.4 of [RFC1034] describes negative response caching, but notes it is optional and only talks about name errors (NXDOMAIN). This is the origin of using the SOA MINIMUM field as a negative caching TTL.

[RFC2308] updated [RFC1034] to specify new requirements for DNS negative caching, including making it mandatory for name error (NXDOMAIN) and no data responses. It further specified optional negative caching for two DNS resolution failure cases: server failure and dead / unreachable servers.

FOR DISCUSSION: RFC 2308 seems to use RFC 2119 keywords somewhat inconsistently when in comes to requirements for negative caching of type (1) and (2) responses. For example:

- *Abstract: "negative caching should no longer be seen as an optional part of..."

- *Section 5: "A negative answer that resulted from a name error (NXDOMAIN) should be cached..."

- *Section 5: "A negative answer that resulted from a no data error (NODATA) should be cached..."

- *Section 8: "Negative caching in resolvers is no-longer optional, if a resolver caches anything it must also cache negative answers."

This document updates [RFC2308] to require negative caching of DNS resolution failures, and provides additional examples of resolution failures.

1.1. Motivation

Operators of DNS services have known for some time that recursive resolvers become more aggressive when they experience resolution failures. A number of different anecdotes, experiments, and incidents support this claim.

[The authors vaguely recall stories of a moderately popular DNSBL that wanted to shut down, but found that not responding or REFUSED caused an overwhelming amount of traffic. Are there any citable references to this happening?]

In December 2009, a secondary server for a number of in-addr.arpa subdomains saw its traffic suddenly double, and queries of type

DNSKEY in particular increase by approximately two orders of magnitude, coinciding with a DNSSEC key rollover by the zone operator [roll-over-and-die]. This predated a signed root zone and an operating system vendor was providing non-root trust anchors to the recursive resolver, which became out-of-date following the rollover. Unable to validate responses for the affected in-addr.arpa zones, recursive resolvers aggressively retried their queries.

In 2016, the internet infrastructure company Dyn experienced a large attack that impacted many high-profile customers. As documented in a technical presentation detailing the attack [dyn-attack], Dyn staff wrote: "At this point we are now experiencing botnet attack traffic and what is best classified as a 'retry storm'. Looking at certain large recursive platforms > 10x normal volume."

In 2018 the root zone key signing key (KSK) was rolled over [root-ksk-roll]. Throughout the rollover period, the root servers experienced a significant increase in DNSKEY queries. Before the rollover, a.root-servers.net and j.root-servers.net together received about 15 million DNSKEY queries per day. At the end of the revocation period, they received 1.2 billion per day -- an 80x increase. Removal of the revoked key from the zone caused DNSKEY queries to drop to post-rollover but pre-revoke levels, indicating there is still a population of recursive resolvers using the previous root trust anchor and aggressively retrying DNSKEY queries.

In 2021, Verisign researchers used botnet query traffic to demonstrate that certain large, public recursive DNS services exhibit very high query rates when all authoritative name servers for a zone return REFUSED or SERVFAIL [botnet]. When configured normally, query rates for a single botnet domain averaged approximately 50 queries per second. However, when configured to return SERVFAIL, the query rate increased to 60,000 per second. Furthermore, increases were also observed at the Root and TLD levels, even though delegations at those levels were unchanged and continued operating normally.

Later that same year, on October 4, Facebook experienced a widespread and well-publicized outage [fb-outage]. During the 6-hour outage, none of Facebook's authoritative name servers were reachable and did not respond to queries. Recursive name servers attempting to resolve Facebook domains experienced timeouts. During this time query traffic on the .COM/.NET infrastructure increased from 7,000 to 900,000 queries per second [CITATION NEEDED].

1.2. Related Work

[RFC2308] describes negative caching for four types of DNS queries and responses: Name errors, no data, server failures, and dead /

unreachable servers. It places the strongest requirements on negative caching for name errors and no data responses, while server failures and dead servers are left as optional.

[RFC4697] is a Best Current Practice that documents observed resolution misbehaviors. It describes a number of situations that can lead to excessive queries from recursive resolvers, including: requerying for delegation data, lame servers, responses blocked by firewalls, and records with zero TTL. [RFC4697] makes a number of recommendations, varying from "SHOULD" to "MUST."

An expired Internet Draft describes "The DNS thundering herd problem" [thundering-herd] as a situation arising when cached data expires at the same time for a large number of users. Although that document is not focused on negative caching, it does describe the benefits of combining multiple, identical queries to upstream name servers. That is, when a recursive resolver receives multiple queries for the same name, class, and type that cannot be answered from cached data, it should combine or join them into a single upstream query, rather than emit repeated, identical upstream queries.

[RFC5452], "Measures for Making DNS More Resilient against Forged Answers," includes a section that describes the phenomenon known as birthday attacks. Here, again, the problem arises when a recursive resolver emits multiple, identical upstream queries. Multiple outstanding queries makes it easier for an attacker to guess and correctly match some of the DNS message parameters, such as the port number and ID field. This situation is only exacerbated in the case of timeout-based resolution failures. DNSSEC, of course, is a suitable defense to spoofing attacks.

[RFC8767] describes "Serving Stale Data to Improve DNS Resiliency." This permits a recursive resolver to return possibly stale data when it is unable to refresh cached, expired data. It introduces the idea of a failure recheck timer and says: "Attempts to refresh from non-responsive or otherwise failing authoritative nameservers are recommended to be done no more frequently than every 30 seconds."

1.3. Terminology

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

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

2. Conditions That Lead To DNS Resolution Failures

A DNS resolution failure occurs when none of the servers available to a resolver client provide any useful response data for a particular query name, type, and class.

It is common for resolvers to have multiple servers from which to choose for a particular query. For example, in the case of stub-to-recursive, the stub resolver may be configured with multiple recursive resolver addresses. In the case of recursive-to-authoritative, a given zone usually has more than one name server (NS record), each of which can have multiple IP addresses and multiple transports.

Nothing in this document prevents a resolver from retrying a query at a different server, or the same server over a different transport. In the case of timeouts, a resolver can retry the same server and transport a limited number of times.

If any one of the available servers provides a useful response, then it is not considered a resolution failure. However, if none of the servers for a given query tuple <name, type, class> provide a useful response, the result is a resolution failure.

Note that NXDOMAIN and NOERROR/NODATA responses are not conditions for resolution failure. In these cases, the server is providing a useful response, either indicating that a name does not exist, or that no data of the requested type exists at the name. These negative responses can be cached as described in [RFC2308].

The remainder of this section describes a number of different conditions that can lead to resolution failure.

2.1. Server Failure

Server failure is defined in [RFC1035] as: "The name server was unable to process this query due to a problem with the name server." A server failure is signaled by setting the RCODE field to SERVFAIL.

Authoritative servers, and more specifically secondary servers, return server failure responses when they don't have any valid data for a zone. That is, a secondary server has been configured to serve a particular zone, but is unable to retrieve or refresh the zone data from the primary server.

Recursive servers return server failure in response to a number of different conditions, including many described below.

2.2. Refused Response Code

A name server returns a message with the RCODE field set to REFUSED when it refuses to process the query for policy reasons.

Authoritative servers generally return REFUSED when processing a query for which they are not authoritative. For example, a server that is configured to be authoritative for only the EXAMPLE.NET zone, may return REFUSED in response to a query for EXAMPLE.COM.

Recursive servers generally return REFUSED for query sources that do not match configured access control lists. For example, a server that is configured to allow queries from only 2001:DB8:1::/48 may return REFUSED in response to a query from 2001:DB8:5::1.

2.3. Timeouts

A timeout occurs when a resolver fails to receive any response from a server within a reasonable amount of time. [RFC2308] refers to this as a "dead / unreachable server."

Note that resolver implementations may have two types of timeouts: a smaller timeout which might trigger a query retry and a larger timeout after which the server is considered unresponsive.

Timeouts can present a particular problem for negative caching, depending on how the resolver handles multiple, outstanding queries for the same <query name, type, class> tuple. For example, consider a very popular website in a zone whose name servers are all unresponsive. A recursive resolver might receive tens or hundreds of queries per second for the popular website. If the recursive server implementation "joins" these outstanding queries together, then it only sends one recursive-to-authoritative query for the numerous pending stub-to-recursive queries. If, however, the implementation does not join outstanding queries together, then it sends one recursive-to-authoritative query for each stub-to-recursive query. If the incoming query rate is high and the timeout is large, this might result in hundreds or thousands of recursive-to-authoritative queries while waiting for an authoritative server to time out.

2.4. Delegation Loops

A delegation loop, or cycle, can occur when one domain utilizes name servers in a second domain, and the second domain uses name servers in the first. For example:

```
FOO.EXAMPLE.     NS      NS1.EXAMPLE.COM.
FOO.EXAMPLE.     NS      NS2.EXAMPLE.COM.

EXAMPLE.COM.     NS      NS1.FOO.EXAMPLE.
EXAMPLE.COM.     NS      NS2.FOO.EXAMPLE.
```

In this example, no names under FOO.EXAMPLE or EXAMPLE.COM can be resolved because of the delegation loop. Note that delegation loop may involve more than two domains. A resolver that does not detect delegation loops may generate DDoS-levels of attack traffic to authoritative name servers, as documented in the TsuNAME vulnerability [TsuNAME].

2.5. Alias Loops

An alias loop, or cycle, can occur when one CNAME or DNAME RR refers to a second name, which in turn is specified as an alias for the first. For example:

```
APP.FOO.EXAMPLE.         CNAME   APP.EXAMPLE.NET.
APP.EXAMPLE.NET.         CNAME   APP.FOO.EXAMPLE.
```

The need to detect CNAME loops has been known since at least [RFC1034] which states in Section 3.6.2:

"Of course, by the robustness principle, domain software should not fail when presented with CNAME chains or loops; CNAME chains should be followed and CNAME loops signaled as an error."

2.6. DNSSEC Validation Failures

Negative caching of DNSSEC validation errors is described in section 4.7 of [RFC4035].

FOR DISCUSSION: RFC4035 says "resolvers MAY cache data with invalid signatures" while in this document all resolution failures MUST be negatively cached. The focus of 4035 seems to be on caching bad *data* rather than caching a more general resolution failure (e.g. inability to retrieve keys).

3. Requirements for Caching Resolution Failures

3.1. Retries and Timeouts

A resolver MUST NOT retry a given query over a server's transport more than twice (i.e., three queries in total) before considering the server's transport unresponsive for that query.

A resolver MAY retry a given query over a different transport to the same server if it has reason to believe the transport is available for that server.

This document does not place any requirements on timeout values, which may be implementation- or configuration-dependent. It is generally expected that typical timeout values range from 3 to 30 seconds.

3.2. Caching

Resolvers MUST implement a cache for resolution failures. The purpose of this cache is to eliminate repeated upstream queries that cannot be resolved. When an incoming query matches a cached resolution failure, the resolver MUST NOT send any corresponding outgoing queries until after the cache entries expire.

Implementation details [requirements?] for such a cache are not specified in this document. The implementation might cache different resolution failure conditions differently. For example, DNSSEC validation failures might be cached according to the queried name, class, and type, whereas unresponsive servers might be cached only according to the server's IP address.

Resolvers MUST cache resolution failures for at least 5 seconds. The value of 5 seconds is chosen as a reasonable amount of time that an end user could be expected to wait.

Resolvers SHOULD employ an exponential backoff algorithm to increase the amount of time for subsequent resolution failures. For example, the initial time for negatively caching a resolution failure is set to 5 seconds. The time is doubled after each retry that results in another resolution failure. Consistent with [RFC2308], resolution failures MUST NOT be cached for longer than 5 minutes.

Notwithstanding the above, resolvers SHOULD implement measures to mitigate resource exhaustion attacks on the failed resolution cache. That is, the resolver should limit the amount of memory and/or processing time devoted to this cache.

3.3. Requerying Delegation Information

Quoting from [RFC4697]:

There can be times when every name server in a zone's NS RRSet is unreachable (e.g., during a network outage), unavailable (e.g., the name server process is not running on the server host), or misconfigured (e.g., the name server is not authoritative for the given zone, also known as "lame").

This document reiterates the requirement from Section 2.1.1 of [RFC4697]:

An iterative resolver MUST NOT send a query for the NS RRSet of a non-responsive zone to any of the name servers for that zone's parent zone. For the purposes of this injunction, a non-responsive zone is defined as a zone for which every name server listed in the zone's NS RRSet:

1. is not authoritative for the zone (i.e., lame), or

2. returns a server failure response (RCODE=2), or

3. is dead or unreachable according to Section 7.2 of [RFC2308].

FOR DISCUSSION: the requirement quoted above may be problematic today. e.g., focusing on NS as the query type (a) probably goes against qname minimization, and (b) is not the real problem. Also RFC 4697 doesn't place any time restriction (TTL) on this.

4. IANA Considerations

None

5. Security Considerations

As noted in Section 3.2, an attacker might attempt a resource exhaustion attack by sending queries for a large number of names and/or types that result in resolution failure. Resolvers SHOULD implement measures to protect themselves and bound the amount of memory devoted to caching resolution failures.

6. Privacy Considerations

This specification has no impact on user privacy.

7. Acknowledgments

The authors wish to thank Mukund Sivaraman, Petr Spacek, and other members of the DNSOP working group for their feedback and contributions.

8. Change Log

RFC Editor: Please remove this section before publication.

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

From -00 to -01:

*use phrase "the initial TTL for negatively caching a resolution failure" instead of "negative cache TTL"

*typos, etc

From dwmtwc-01 to ietf-00:

*Adopted by WG

From -00 to -01:

*Clarify retries and timeouts to apply on a per-query basis.

*Say more about the 5 second caching requirement in TTLs section.

*Expanded opening paragraphs of section 2, now titled "Conditions That Lead To DNS Resolution Failures".

*Text from the former section 3.3 ("Scope") moved to top of section 2.

*Section 3.2 was formerly "TTLs" and is now "Caching". The draft no longer requires e.g. caching by tuples, but now just requires caching failures so that repeated queries are not sent out.

*State that resolvers should protect themselves from cache resource exhaustion attacks.

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>.

[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>.

[RFC4697] Larson, M. and P. Barber, "Observed DNS Resolution Misbehavior", BCP 123, RFC 4697, DOI 10.17487/RFC4697, October 2006, <https://www.rfc-editor.org/info/rfc4697>.

[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>.

[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

[botnet] Wessels, D. and M. Thomas, "Botnet Traffic Observed at Various Levels of the DNS Hierarchy", May 2021, <https://indico.dns-oarc.net/event/38/contributions/841/>.

[dyn-attack] Sullivan, A., "Dyn, DDoS, and DNS", March 2017, <https://ccnso.icann.org/sites/default/files/file/field-file-attach/2017-04/presentation-oracle-dyn-ddos-dns-13mar17-en.pdf>.

[fb-outage] Janardhan, S., "More details about the October 4 outage", October 2021, <https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/>.

[RFC0882] Mockapetris, P., "Domain names: Concepts and facilities", RFC 882, DOI 10.17487/RFC0882, November 1983, <https://www.rfc-editor.org/info/rfc882>.

[RFC0883] Mockapetris, P., "Domain names: Implementation specification", RFC 883, DOI 10.17487/RFC0883, November 1983, <https://www.rfc-editor.org/info/rfc883>.

[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>.

[RFC5452] Hubert, A. and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers", RFC 5452, DOI 10.17487/RFC5452, January 2009, <https://www.rfc-editor.org/info/rfc5452>.

[RFC8767] Lawrence, D., Kumari, W., and P. Sood, "Serving Stale Data to Improve DNS Resiliency", RFC 8767, DOI 10.17487/

RFC8767, March 2020, <https://www.rfc-editor.org/info/rfc8767>.

[roll-over-and-die] Michaleson, G., Wallström, P., Arends, R., and G. Huston, "Roll Over and Die?", February 2010, <https://www.potaroo.net/ispcol/2010-02/rollover.html>.

[root-ksk-roll] Müller, M., Thomas, M., Wessels, D., Hardaker, W., Chung, T., Toorop, W., and R.v. Rijswijk-Deij, "Roll, Roll, Roll Your Root: A Comprehensive Analysis of the First Ever DNSSEC Root KSK Rollover", October 2019, <https://dl.acm.org/doi/10.1145/3355369.3355570>.

[thundering-herd] Sivaraman, M. and C. Liu, "The DNS thundering herd problem (expired Internet Draft)", June 2020, <https://datatracker.ietf.org/doc/draft-muks-dnsop-dns-thundering-herd/>.

[TsuNAME] Moura, G. C. M., Castro, S., Heidemann, J., and W. Hardaker, "TsuNAME: exploiting misconfiguration and vulnerability to DDoS DNS", November 2021, <https://dl.acm.org/doi/10.1145/3487552.3487824>.

Authors' Addresses

Duane Wessels
Verisign
12061 Bluemont Way
Reston

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

William Carroll
Verisign
12061 Bluemont Way
Reston

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

Matthew Thomas
Verisign
12061 Bluemont Way
Reston

Phone: +1 703 948-3200
Email: mthomas@verisign.com

URI: https://verisign.com