DNSOP Working Group
Internet-Draft
Updates: 1034, 1035 (if approved)
Intended status: Standards Track
Expires: December 29, 2017

D. Lawrence
Akamai Technologies
W. Kumari
Google

June 27, 2017

Serving Stale Data to Improve DNS Resiliency
draft-tale-dnsop-serve-stale-01

Abstract

This draft defines a method for recursive resolvers to use stale DNS data to avoid outages when authoritative nameservers cannot be reached to refresh expired data.

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/vttale/serve-stale>. 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 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 December 29, 2017.

Table of Contents

1. Introduction

Traditionally the Time To Live (TTL) of a DNS resource record has been understood to represent the maximum number of seconds that a record can be used before it must be discarded, based on its description and usage in [RFC1035] and clarifications in [RFC2181]. Specifically, [RFC1035] Section 3.2.1 says that it "specifies the time interval that the resource record may be cached before the source of the information should again be consulted".

Notably, the original DNS specification does not say that data past its expiration cannot be used. This document proposes a method for how recursive resolvers should handle stale DNS data to balance the competing needs of resiliency and freshness. It is predicated on the observation that authoritative server unavailability can cause outages even when the underlying data those servers would return is typically unchanged.

There are a number of reasons why an authoritative server may become unreachable, including Denial of Service (DoS) attacks, network issues, and so on. This document suggests that, if the recursive

server is unable to contact the authoritative server but still has data for the query name, it essentially extends the TTL of the existing data on the assumption that "stale bread is better than no bread".

Several major recursive resolver operations currently use stale data for answers in some way, including Akamai, OpenDNS, and Xerocole.

2. Terminology

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].

For a comprehensive treatment of DNS terms, please see [RFC7719].

3. Description

Three notable timers drive considerations for the use of stale data, as follows:

- A client response timer, which is the maximum amount of time a recursive resolver should allow between the receipt of a resolution request and sending its response.

- A query resolution timer, which caps the total amount of time a recursive resolver spends processing the query.

- A maximum stale timer, which caps the amount of time that records will be kept past their expiration.

Recursive resolvers already have the second timer; the first and third timers are new concepts for this mechanism.

When a request is received by the recursive resolver, it SHOULD start the client response timer. This timer is used to avoid client timeouts. It SHOULD be configurable, with a recommended value of 1.8 seconds.

The resolver then checks its cache for an unexpired answer. If it finds none and the Recursion Desired flag is not set in the request, it SHOULD immediately return the response without consulting the cache for expired records.

If iterative lookups will be done, it SHOULD start the query resolution timer. This timer bounds the work done by the resolver, and is commonly around 10 to 30 seconds. [BIND 9 used to use a hard-coded constant of 30 seconds and has more recently added a

configuration parameter that defaults to 10 seconds and is capped at 30. A rigorous exploration of other implementations has not yet been done.]

If the answer has not been completely determined by the time the client response timer has elapsed, the resolver SHOULD then check its cache to see whether there is expired data that would satisfy the request. If so, it adds that data to the response message and SHOULD set the TTL of each expired record in the message to 1 second. The response is then sent to the client while the resolver continues its attempt to refresh the data. 1 second was chosen because historically 0 second TTLs have been problematic for some implementations. It not only sidesteps those potential problems with no practical negative consequence, it would also rate limit further queries from any client that is honoring the TTL, such as a forwarding resolver.

The maximum stale timer is used for cache management and is independent of the query resolution process. This timer is conceptually different from the maximum cache TTL that exists in many resolvers, the latter being a clamp on the value of TTLs as received from authoritative servers. The maximum stale timer SHOULD be configurable, and defines the length of time after a record expires that it SHOULD be retained in the cache. The suggested value is 7 days, which gives time to notice the resolution problem and for human intervention for fixing it.

This same basic technique MAY be used to handle stale data associated with delegations. If authoritative server addresses are not able to be refreshed, resolution can possibly still be successful if the authoritative servers themselves are still up.

4. Implementation Caveats

Answers from authoritative servers that have a DNS Response Code of either 0 (NOERROR) or 3 (NXDOMAIN) MUST be considered to have refreshed the data at the resolver. In particular, this means that this method is not meant to protect against operator error at the authoritative server that turns a name that is intended to be valid into one that is non-existent, because there is no way for a resolver to know intent.

[Paul Vixie has suggested that it be made explicit that an auth NXDOMAIN cause all data, even stale data, below the NXDOMAIN to also be removed, a la https://datatracker.ietf.org/doc/draft-vixie-dnsext-resimprove/. Conceptually this certainly has its appeal but addressing it in this document when resimprove has not progressed has procedural problems. This paragraph will be removed in the next

draft, either dropping the idea here completely or blessing it based on positive feedback to do so.]

Resolution is given a chance to succeed before stale data is used to adhere to the original intent of the design of the DNS. This mechanism is only intended to add robustness to failures, and to be enabled all the time. If stale data were used immediately and then a cache refresh attempted after the client response has been sent, the resolver would frequently be sending data that it would have had no trouble refreshing.

It is important to continue the resolution attempt after the stale response has been sent, until the query resolution timeout, because some pathological resolutions can take many seconds to succeed as they cope with unavailable servers, bad networks, and other problems. Stopping the resolution attempt when the response with expired data has been sent would mean that answers in these pathological cases would never be refreshed.

Canonical Name (CNAME) records mingled in the expired cache with other records at the same owner name can cause surprising results. This was observed with an initial implementation in BIND, where a hostname changed from having an IPv4 Address (A) record.to a CNAME. The version of BIND being used did not evict other types in the cache when a CNAME was received, which in normal operations is not a significant issue. However, after both records expired and the authorities became unavailable, the fallback to stale answers returned the older A instead of the newer CNAME.

[This probably applies to other occluding types, so more thought should be given to the overall issue.]

Keeping records around after their normal expiration will of course cause caches to grow larger than if records were removed at their TTL. Specific guidance on managing cache sizes is outside the scope of this document. Some areas for consideration include whether to track the popularity of names in client requests versus evicting by maximum age, and whether to provide a feature for manually flushing only stale records.

5. Implementation Status

[RFC Editor: per RFC 6982 this section should be removed prior to publication.]

The algorithm described in this draft was originally implemented as a patch to BIND 9.7.0. It has been in production on Akamai's production network since 2011, and effectively smoothed over

transient failures and longer outages that would have resulted in major incidents. The patch has been contributed to the Internet Systems Consortium in anticipation that it will be incorporated to their main BIND distribution.

Unbound has a similar feature for serving stale answers, but it works in a very different way by returning whatever cached answer it has before trying to refresh expired records.

6. Security Considerations

The most obvious security issue is the increased likelihood of DNSSEC validation failures when using stale data because signatures could be returned outside their validity period. This would only be an issue if the authoritative servers are unreachable, the only time the techniques in this document are used, and thus does not introduce a new failure in place of what would have otherwise been success.

Additionally, bad actors have been known to use DNS caches to keep records alive even after their authorities have gone away. This makes that easier.

7. Privacy Considerations

This document does not add any practical new privacy issues.

8. NAT Considerations

The method described here is not affected by the use of NAT devices.

9. IANA Considerations

This document contains no actions for IANA. This section will be removed during conversion into an RFC by the RFC editor.

10. Acknowledgements

The authors wish to thank Matti Klock, Mukund Sivaraman, Jean Roy, and Jason Moreau for initial review.

11. References

11.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc2181>.

11.2. Informative References

[RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <http://www.rfc-editor.org/info/rfc7719>.

Authors' Addresses

David C Lawrence
Akamai Technologies
150 Broadway
Cambridge MA 02142-1054
USA

Email: tale@akamai.com

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View CA 94043
USA

Email: warren@kumari.net