From: **David C Lawrence** tale@akamai.com
Subject: Serve Stale patch
Date: February 28, 2017 at 3:59 PM
To: Victoria Risk vicky@isc.org, Stephen Morris stephen@isc.org, Mukund Sivaraman muks@isc.org, Evan Hunt each@isc.org

DC

Here it is, the long awaited patch! There are two attachments, the first being the patch and the second the current version of the Internet-Draft that I'm waiting to submit after Warren is done with the editing pen.

Some things to note about the patch:

* Per the comment in the draft about not evicting CNAMEs in the cache when other data arrives, this can result in unexpected behaviour once everything goes stale and the CNAME comes back into play after new authoritative data had changed the zone. We had an incident related to this. This has not been addressed in the patch; if I had, I was leaning in the direction of checking for an existing CNAME conflict when adding new data and evicting the old data.

* This does not handle using stale glue really, which is a shame. I believe it should, but I just didn't get into messing around with the adb. Personally I think if you have a stale delegation for example.com you should still be able to use it to resolve names.

* There's some work in there related to reloading the dump file, which I realize was meant only for testing and not a production feature even before this came along. We had a thought that this would also improve generalized resiliency to preserve data across restarts, but since the dump load doesn't have a provision for loading negative answers I didn't finish that. The timestamps in the dump file are still written to reflect stale age though, which could be really surprising for someone looking at it and seeing much longer TTLs than they expect.

Sorry again that this took so long, but I hope that it is useful for you.

DNSOP Working Group — D. Lawrence
Internet-Draft — Akamai Technologies
Intended status: Standards Track — W. Kumari
Expires: August 26, 2017 — Google
February 22, 2017

Serving Stale Data to Improve DNS Resiliency
draft-tale-dnsop-serve-stale-00

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.

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

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 August 26, 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.

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

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 expired record cache.

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 30 seconds.

If the answer has not been completely determined when 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 sends a response and the TTL fields of the expired records SHOULD be set to 1.

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 problem and and for human intervention for fixing it.

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.

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 not be a standard operational occurrence as would happen if stale data were used immediately and then a cache refresh attempted after the client response has been sent.

It is important to continue the resolution attempt after the stale response has been sent because some pathological resolutions can take at least a dozen seconds succeed as they cope with down servers, bad networks, and other problems. Stopping the resolution attempt when the response 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 a CNAME record to an IPv4 Address (A) record. BIND does not evict CNAMEs in the cache when other types are received, which in normal operations is not an issue. However, after both records expired and the authorities became unavailable, the fallback to stale answers returned the older CNAME instead of the newer A.

(This might apply 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.

5. 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.

Additionally, bad actors have been known to use DNS caches as a kind of perpetual cloud database, keeping records alive even after their authorities have gone away. This makes that easier.

6. Privacy Considerations

This document does not add any practical new privacy issues.

7. NAT Considerations

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

8. IANA Considerations

This document contains no actions for IANA.

9. Acknowledgements

The authors wish to thank Matti Klock for initial review.

10. 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>.

[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