Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: March 11, 2021

S. Huque
Salesforce
P. Vixie
Farsight Security
R. Dolmans
NLnet Labs

September 7, 2020

Delegation Revalidation by DNS Resolvers
draft-ietf-dnsop-ns-revalidation-00

Abstract

This document recommends improved DNS [RFC1034] [RFC1035] resolver behavior with respect to the processing of Name Server (NS) resource record sets (RRset) during iterative resolution. When following a referral response from an authoritative server to a child zone, DNS resolvers should explicitly query the authoritative NS RRset at the apex of the child zone and cache this in preference to the NS RRset on the parent side of the zone cut. Resolvers should also periodically revalidate the child delegation by re-quering the parent zone at the expiration of the TTL of the parent side NS RRset.

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 March 11, 2021.

Table of Contents

1. Introduction

RFC EDITOR: PLEASE REMOVE THIS PARAGRAPH BEFORE PUBLISHING: The source for this draft is maintained in GitHub at: https://github.com/shuque/ns-revalidation

This document recommends improved DNS resolver behavior with respect to the processing of NS record sets during iterative resolution. The first recommendation is that resolvers, when following a referral response from an authoritative server to a child zone, should explicitly query the authoritative NS RRset at the apex of the child zone and cache this in preference to the NS RRset on the parent side of the zone cut. The second recommendation is to revalidate the delegation by re-quering the parent zone at the expiration of the TTL of the parent side NS RRset.

2. Motivation

There is wide variability in the behavior of deployed DNS resolvers today with respect to how they process delegation records. Some of them prefer the parent NS set, some prefer the child, and for others, what they preferentially cache depends on the dynamic state of queries and responses they have processed. This document aims to bring more commonality and predictability by standardizing the behavior in a way that comports with the DNS protocol.

The delegation NS RRset at the bottom of the parent zone and the apex NS RRset in the child zone are unsynchronized in the DNS protocol. [RFC1034] Section 4.2.2 says "The administrators of both zones should insure that the NS and glue RRs which mark both sides of the cut are consistent and remain so.". But for a variety of reasons they could not be. Officially, a child zone's apex NS RRset is authoritative and thus has a higher cache credibility than the parent's delegation NS RRset, which is non-authoritative glue ([RFC2181], Section 5.4.1. "Ranking data", and Section 6.1. "Zone authority"). Hence the NS RRset "below the zone cut" should immediately replace the parent's delegating NS RRset in cache when an iterative caching DNS resolver crosses a zone boundary. However, this can only happen if (1) the resolver receives the authoritative NS RRset in the Authority section of a response from the child zone, which is not mandatory, or (2) if the resolver explicitly issues an NS RRset query to the child zone as part of its iterative resolution algorithm. In the absence of this, it is possible for an iterative caching resolver to never learn the authoritative NS RRset for a zone, unless a downstream client of the resolver explicitly issues such an NS query, which is not something that normal enduser applications do, and thus cannot be relied upon to occur with any regularity.

Increasingly, there is a trend towards minimizing unnecessary data in DNS responses. Several popular DNS implementations default to such a configuration (see "minimal-responses" in BIND and Unbound). So, they may never include the authoritative NS RRset in the Authority

section of their responses.

A common reason that zone owners want to ensure that resolvers place the authoritative NS RRset preferentially in their cache is that the TTLs may differ between the parent and child side of the zone cut. Some DNS Top Level Domains (TLDs) only support long fixed TTLs in their delegation NS sets. In fact, the Extensible Provisioning Protocol (EPP) [RFC5731], that is often used by TLDs to configure delegation parameters has no provision to set the TTL. This inhibits a child zone owner's ability to make more rapid changes to their nameserver configuration using a shorter TTL, if resolvers have no systematic mechanism to observe and cache the child NS RRset.

A child zone's delegation still needs to be periodically revalidated at the parent to make sure that the parent zone has not legitimately re-delegated the zone to a different set of nameservers, or even removed the delegation. Otherwise, resolvers that refresh the TTL of a child NS RRset on subsequent queries or due to pre-fetching, may cling to those nameservers long after they have been re-delegated elsewhere. This leads to the second recommendation in this document, "Delegation Revalidation" - Resolvers should record the TTL of the parent's delegating NS RRset, and use it to trigger a revalidation action.

3. Upgrading NS RRset Credibility

* When a delegation response is received during iteration, a validation query should be sent in parallel with the resolution of the triggering query, to the delegated nameservers for the newly discovered zone cut. Note that validating resolvers today, when following a secure referral, already need to dispatch a query to the delegated nameservers for the DNSKEY RRset, so this validation query could be sent in parallel with that DNSKEY query.

* A validation query consists of a query for the child's apex NS RRset, sent to the newly discovered delegation's nameservers. Normal iterative logic applies to the processing of responses to validation queries, including storing the results in cache, trying the next server on SERVFAIL or timeout, and so on. Positive answers to this validation query will be cached with an authoritative data ranking. Successive queries directed to the same zone will be directed to the nameservers listed in the

child's apex, due to the ranking of this answer. If the validation query fails, the parent NS RRset will remain the one with the highest ranking and will be used for successive queries.

* Some resolvers may choose to delay the response to the triggering query until both the triggering query and the validation query have been answered. In practice, we expect many implementations may answer the triggering query in advance of the validation query for performance reasons. An additional reason is that there are number of nameservers in the field that (incorrectly) fail to answer explicit queries for NS records, and thus the revalidation logic may need to be applied lazily and opportunistically to deal with them.

* If the resolver chooses to delay the response, and there are no nameserver names in common between the child's apex NS RRset and the parent's delegation NS RRset, then the responses received from forwarding the triggering query to the parent's delegated nameservers should be discarded after validation, and this query should be forwarded again to the child's apex nameservers.

4. Delegation Revalidation

This documents proposes two mechanisms to perform delegation revalidation: an extensive and a simple mechanism. [TODO: should we keep just the simple mechanism?]

The extensive mechanism:

* The lowest TTL found in a parent zone's delegating NS RRset should be stored in the cache and used to trigger delegation revalidation as follows: Whenever a cached RRset is being considered for use in a response, the cache should be walked upward toward the root, looking for expired delegations. At the first expired delegation encountered while walking upward toward the root, revalidation should be triggered, putting the processing of dependent queries

on hold until validation is complete.

* To revalidate a delegation, the iterative caching DNS resolver will redo resolution of the triggering query at the closest enclosing zone cut above the revalidation point. If Query-name Minimization [RFC7816] is being employed, this may not be the full name of the triggering query, but the query name with some number of left most labels excised as dictated by the qname minimization algorithm. While searching for these nameservers, additional revalidations may occur, perhaps placing a chain of dependent queries on hold, unwinding in downward order as revalidations closer to the root must be complete before revalidations further from the root can begin.

* If a delegation can be revalidated at the same node, then the old apex NS RRset should be deleted from cache and then the new delegating NS RRset should be stored in cache. The minimum TTL from the new delegating NS RRset should also be stored in cache to facilitate future revalidations. This order of operations ensures that the RRset credibility rules do not prevent the new delegating NS RRset from entering the cache. It is expected that the child's apex NS RRset will rapidly replace the parent's delegating NS RRset as soon as iteration restarts after the revalidation event.

* If the new delegating NS RRset cannot be found (RCODE=NXDOMAIN) or if there is a new zone cut at some different level of the hierarchy (insertion or deletion of a delegation point above the revalidation point) or if the new RRset shares no nameserver names in common with the old one (indicating some kind of redelegation, which is rare) then the cache should be purged of all names and RRsets at or below the revalidation point. This facilitates redelegation or revocation of a zone by a parent zone administrator, and also conserves cache storage by deleting unreachable data.

The simple mechanism:

* Cap the time to cache the child NS RRset to the lower of child and parent NS RRset TTL. The normal iterative resolution algorithm

will then cause delegation revalidation to naturally occur at the expiration of the capped child NS TTL, along with dispatching of the validation query to upgrade NS RRset credibility.

4.1. Using the DS Record TTL

If both parent and child zone are DNSSEC [RFC4033] [RFC4034] [RFC4035] signed with a corresponding secure delegation between them, then expiration of the Delegation Signer (DS) record set will cause revalidation of the current child zone's DNSKEY set. According to RFC 4035, Section 2.4, "The TTL of a DS RRSet SHOULD match the TTL of the delegating NS RRset", so this revalidation should be triggered on the same time scale, and thus responses from the stale child nameservers would no longer be trusted. However, delegation revalidation is still necessary to locate the current nameserver addresses to which subsequent DNS queries should be directed.

In practice, the DS TTL often differs from the delegating NS TTL. For example, currently the root zone and the COM and NET TLDs all set an NS RRset TTL of 2 days, while the DS RRset TTL is just 1 day. RSSAC-003 [rssac-003] makes the following observation: "In the root zone, delegating NS records have a 2 day TTL. However, the DS records have a 1 day TTL, against the advice of RFC 4035. This is not particularly surprising since a mistake with a DS record can deny resolution for all names under a TLD. Given the way that DS records are currently used in the root zone (e.g., usually matching just one TLD KSK) it is better for them to have a lower TTL in the event of an emergency change."

If a secure delegation is present, resolvers may use the DS RRset's TTL as the revalidation interval in preference to to the delegating NS RRSet TTL. (Question: should this be the recommendation instead?; after all the DS is signed so its TTL cannot be spoofed.?)

5. Optimizations

TODO: mention possible optimizations: record whether certain nameservers return the child NS set in their authoritative section responses and subsequently forego the extra child NS query for a period of time. Suggest that authoritative servers that do minimal responses return their NS sets in response to DNSKEY queries, and resolvers that see such behavior may also subsequently forego the

extra child NS query (from Olafur G.).

6. Re-delegations and Delegation Removals

TODO: mention in more detail what to do when resolvers observe redelegations or removal of delegations at the parent. A quick initial summary follows.

If a delegation is removed (i.e. the parent returns NXDOMAIN), then cache contents should be treated as described in [RFC8020] -- ideally the resolver's cache should be entirely pruned at the delegation point, or the cached contents below the delegation may be allowed to be used until they expire. Similar treatment of the cache should be followed if the child zone has been entirely re-delegated to a new set of nameservers. If only a subset of nameservers have been re-delegated, then no new cache cleanup action is needed.

7. IANA Considerations

This document includes no request to IANA.

8. Security Considerations

Upgrading NS RRset Credibility (Section 3) allows resolvers to cache and utilize the authoritative child apex NS RRset in preference to the non-authoriative parent NS RRset. However, it is important to implement the steps described in Delegation Revalidation (Section 4) at the expiration of the parent's delegating TTL. Otherwise, the operator of a malicious child zone, originally delegated to, but subsequently delegated away from, can cause resolvers that refresh TTLs on subsequent NS set queries, or that pre-fetch NS queries, to never learn of the redelegated zone. This problem has been seen in the wild [include reference to Ghost Domains paper here].

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>.

[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>.

[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>.

9.2. Informative References

[I-D.vixie-dnsext-resimprove]
Vixie, P., Joffe, R., and F. Neves, "Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness", Work in Progress, Internet-Draft, draft-vixie-dnsext-resimprove-00, June 23, 2010, <https://tools.ietf.org/html/draft-vixie-dnsext-resimprove-00>.

[I-D.wijngaards-dnsext-resolver-side-mitigation]
Wijngaards, W., "Resolver side mitigations", Work in Progress, Internet-Draft, draft-wijngaards-dnsext-resolver-side-mitigation-01, February 24, 2009, <https://tools.ietf.org/html/draft-wijngaards-dnsext-resolver-side-mitigation-01>.

[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>.

[RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <https://www.rfc-editor.org/info/rfc5731>.

[rssac-003] RSSAC_Caucus, "RSSAC003 Report on Root Zone TTLs", August 2015, <https://www.icann.org/en/system/files/files/rssac-003-root-zone-ttls-21aug15-en.pdf>.

Acknowledgements

Wouter Wijngaards proposed explicitly obtaining authoritative child NS data in [I-D.wijngaards-dnsext-resolver-side-mitigation]. This behavior has been implemented in the Unbound DNS resolver via the "harden-referral-path" option. The combination of child NS fetch and revalidating the child delegation was originally proposed in [I-D.vixie-dnsext-resimprove], by Vixie, Joffe, and Neves.

Authors' Addresses

Shumon Huque
Salesforce

Email: shuque@gmail.com

Paul Vixie
Farsight Security

Email: paul@redbarn.org

Ralph Dolmans
NLnet Labs

Email: ralph@nlnetlabs.nl