DNSEXT Working Group
INTERNET-DRAFT
<draft-vixie-dnsext-resimprove-00.txt>
Intended Status: For Your Information

P. Vixie, ISC
R. Joffe, Centergate
F. Neves, Registro

June 22, 2010

Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness

Status of this Memo

Abstract

This document describes several mechanisms which can be employed by iterative caching DNS resolvers to improve resiliency, robustness, and responsiveness. These improvements are optional and they require no changes to the protocol, or to authority servers, or to DNS stub resolver clients.

1. Introduction

1.1. Iterative caching DNS resolvers can be both compliant and interoperable without also being optimal. Indeed a wide range of optimizations, mechanisms, and outright "tricks" can be employed without affecting correctness or interopability. Such optimizations could be recommended but never required.

1.2. This document describes several practices which can improve the resilience, robustness, and responsiveness of iterative caching DNS resolvers. These practices are not required for correctness or interoperability, and no other DNS protocol agent need be modified to gain the prospective benefits of implementing these practices.

1.3. Three practices are described here:

A. Revalidating a delegation when a parent NS RRset TTL expires.
B. Stopping a downward cache search when an NXDOMAIN is encountered.
C. Upgrading the credibility of NS RRsets upon delegation events.

These practices are described in detail in later sections of this document. While they are described together this single document for editorial convenience, and are known to work well together, they are in no way interdependent.

2. Delegation Revalidation Upon NS RRSet Expiry

2.1. Because the delegating NS RRset at the bottom of the parent zone and the apex NS RRset in the child zone are unsynchronized, the TTL of the parent's delegating NS RRset is meaningless. A child zone's apex NS RRset is authoritative and thus has a higher cache credibility than the parent's delegating NS RRset, so, the NS RRset "below the cut" immediately replaces the parent's delegating NS RRset in cache when an iterative caching DNS resolver crosses a zone cut.

2.2. 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.

2.3. To revalidate a delegation, the iterative caching DNS resolver will forward the query that triggered revalidation to the nameservers at the closest enclosing zone cut above the revalidation point. While searching for these nameservers, additional revalidations may occur, perhaps placing an entire 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.

2.4. 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.

2.5. 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.

2.6. To make the timing of a revalidation event unpredictable from the point of view of a potential cache-spoof attacker, the parent's delegating NS RRset TTL should be reduced by a random fraction of its value before being stored for use in revalidation activities.

3. Stopping Downward Cache Search on NXDOMAIN

3.1. In virtually all existing resolvers, a cached NXDOMAIN is not considered "proof" that there can be no child domains underneath. This is due to an ambiguity in RFC 1034 that failed to distinguish

empty nonterminal domain names from nonexistent names. For DNSSEC, the IETF had to distinguish this case, but the implication on non-DNSSEC resolvers wasn't fully realized.

3.2. When searching downward in its cache, an iterative caching DNS resolver should stop searching if it encounters a cached NXDOMAIN. The response to the triggering query should be NXDOMAIN.

3.3. When an iterative caching DNS resolver stores an NXDOMAIN in its cache, all names and RRsets at or below that node should be deleted since they will have become unreachable.

3.4. By implication, a stream of queries FOO.TLD, BAR.FOO.TLD where FOO.TLD does not exist would normally cause both queries to be forwarded to TLD's nameservers. Following this recommended practice, the second query and indeed any other query for names at or below FOO.TLD would not be forwarded.

4. Upgrading NS RRset Credibility Upon Delegaton Events

4.1. Noting that a parent's delegating NS RRset is nonauthoritative "glue" whereas a child's apex NS RRset is authoritative, the latter will replace the former in cache whenever the latter is encountered. However, it is not mandatory for the child zone's nameservers to include the apex NS RRset in responses, thus it is possible for an iterative caching DNS resolver to never learn the authoritative NS RRset for a zone.

4.2. When a delegation response is received during iteration, a validation query should be sent in parallel with the forwarding of the triggering query to the delegated nameservers for the newly discovered zone cut. The response to the triggering query should be delayed until both the forwarded query and the validation query have been answered.

4.3. 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, propagating NXDOMAIN back to the triggering query, trying the next server on SERVFAIL or timeout, and so on.

4.4. If 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.

5. Security Considerations

5.1. A successful cache exploit which inserted a fake NXDOMAIN can deny more service from an iterative caching DNS resolver that implements the recommendation to stop downward searches when a cached NXDOMAIN is encountered.

5.2. A perfectly timed cache exploit which inserted a fake NS RRset during a delegation validation event could cause one fewer good response to be heard (per TTL expiry interval) from an iterative caching DNS resolver that implements the recommendation to validate delegations.

IANA Considerations

None.

Normative References

[RFC1035] P. Mockapetris, "Domain Names - Implementation and Specification," RFC 1035, USC/Information Sciences Institute, November 1987.

Authors' Addresses

Paul Vixie

Internet Systems Consortium
950 Charter Street
Redwood City, CA, USA
EMail: vixie@isc.org

Rodney Joffe

Centergate Research Group, LLC
420 S Smith Rd
Tempe, AZ 85281 USA
EMail: rjoffe@centergate.com

Frederico A. C. Neves

NIC.br / Registro.br
Av. das Nacoes Unidas, 11541, 7
Sao Paulo, SP 04578-000 BR
EMail: fneves@registro.br