Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 4, 2015

W. Kumari, Ed.
Google
P. Hoffman, Ed.
VPN Consortium

July 3, 2014

Securely Distributing the DNS Root
draft-wkumari-dnsop-dist-root-01

Abstract

This document recommends that recursive DNS resolvers get copies of the root zone, validate it using DNSSEC, populate their caches with the information, and also give negative responses from the validated zone.

[[Note: This document is largely a discussion starting point.]]

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

One of the main advantages of a DNSSEC-signed root zone is that it doesn't matter where you get the data from, as long as you validate the contents of the zone using DNSSEC information. From that point on, you know all of the contents of the root zone at the time that you retrieve and validated the zone.

When a typical recursive resolver starts up, it has an empty cache, the addresses of the root servers. As it begins answering queries, it populates its cache by making a number of queries to the set of root servers, and caching the results. All queries for root zone names that come to the recursive resolver that are not in either its positive or negative cache are sent to one of the root servers. This process cause a large number of the queries that hit the root are so called "junk" queries, such as queries for second-level domains in non-existent TLDs.

This document is describes a mechanism to populate caches in recursive resolvers with the verified contents of the full root zone so that the recursive resolvers have the all of root zone content cached. This technique can be viewed as pre-populating a resolver's cache with the root zone information by retrieving a signed copy of the root zone and verifying the contents.

The two goals of this mechanism are to provide faster negative responses to stub resolver queries that contain junk queries, and to reduce the number of junk queries sent to the root servers. The

mechanism has other minor advantages, but those two are the focus of this document.

1.1. Requirements notation

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].

2. Requirements

In the discussion below, the term "legacy operation" means the way that a recursive resolver acts when it is not using the mechanism describe in this document, namely as a normal validating recursive resolver with no other special features.

In order to implement the mechanism described in this document, a recursive resolver MUST support DNSSEC, and MUST have an up-to-date copy of the DNS root key.

A recursive resolver using this mechanism MUST follow these steps at startup or after clearing its cache:

1. The resolver determines the list of root zone delivery servers. The delivery mechanism is not yet defined in this document, and some possible options for it are described in Section 3.

2. The resolver SHOULD randomly sort the list of zone delivery servers so that all the servers get a fairly even distribution of queries.

3. The resolver SHOULD attempt to transfer the signed root zone using the transfer protocol from each one of the servers until either success is achieved or the list has been exhausted. The resolver MAY attempt to transfer in parallel to minimize startup latency. If the root zone cannot be transferred, the resolver logs this as an error, and MUST fall back to legacy operation.

4. The resolver MUST validate the records in the zone using DNSSEC. If any of the records do not validate, the resolver MUST discard all records received, MUST log an error, and SHOULD try the next server in the list. If no transferred copy of the root zone can be validated, the resolver logs this as an error, and falls back to legacy operation. Note that the resolver MUST validate all of the zone contents, and MUST NOT start using the new contents until all have been validated; the resolver MUST NOT use "lazy validation". This means that the addition of the zone data MUST be an atomic operation.

The resolver MAY store the contents of the validated root zone to disk. If the resolver has a stored copy of the root zone, and the data in the zone is not expired, and that copy was written within the refresh time listed in the zone, the resolver MAY load that zone instead of transferring.

Once the resolver has transferred and validated the zone, it MUST attempt to keep its copy of the root zone up to date. This includes following the refresh, retry, expire logic, with certain modifications:

o If the zone expires (for example, because it cannot retransfer because of blocked TCP connections), the resolver MUST fall back to legacy operation and MUST log an error. It MUST NOT return SERVFAIL to queries only due to its copy of the root zone being expired.

o The resolver MUST validate the contents of the records in the zone using DNSSEC for every transfer. The resolver SHOULD try alternate servers if the validation fails. If the resolver is unable to transfer a copy of the zone that validates, it MUST treat this as an error, MUST discard the received records, and MUST fail back to legacy operation. Note that the resolver MUST validate all of the zone contents, and MUST NOT start using the new contents until all have been validated; the resolver MUST NOT use "lazy validation". This means that the replacement of the current zone data MUST be an atomic operation.

o The resolver SHOULD attempt to restart this process at every retry interval for the root zone.

o The resolver MUST set the AD bit on responses to queries for records in the root zone. This action is the same as if it had inserted the entry into its cache through a "normal" query that received a DNSSEC-validated answer.

o The resolver MUST set the TTL on responses in the same fashion as it would in legacy operation. The difference here is that, when the TTL times out, instead of fetching the new answer from the root, the resolver simply starts the TTL at the maximum listed in the root zone.

Compliant nameservers software MUST include an option to securely cache the root zone (an example name for this option could be "transfer-and-validate-root [yes|no]"). That is, the mechanism described in this document MUST be optional, and the cache operator MUST be able to turn it off and on.

3. Open Question: How Should the Root Zone Be Distributed?

The signed root zone can be distributed over almost any protocol. Because the zone is signed, the distribution protocol does not need to be authenticated. Suggestions for the distribution mechanism include:

AXFR zone transfer within the DNS

HTTP, most likely with appropriately-tuned caching

FTP

[[Others...]]

Note that with any of these methods, the zone does not need to be transferred from the root servers themselves. Instead, a simple discovery mechanism can be built into the protocol that lets a recursive resolver discover where there are servers that will let it transfer the root zone.

4. Open Question: Should Responses Have the AA Bit Set?

A recursive resolver that has a securely validated copy of the root can be thought of in at least two ways: as a smarter cache, or as a pseudo-slave server for the root. This section discusses the ramifications of those two choices. In both scenarios, the resolver will send back NXDOMAIN responses for junk queries without sending queries to the root and the resolver will set the AD bit on the responses. However, the two scenarios differ in whether or not the responses have the AA bit set.

A smarter cache does not set the AA bit. The responses for any query for a name in the root or an NXDOMAIN that is being sent because the TLD is junk come back with the AD bit set but the AA bit not set, just as it would in legacy operation.

A pseudo-slave to the root sets the AA bit in response to any query for a name in the root or an NXDOMAIN that is being sent because the TLD is junk. The reason that this is called a pseudo-slave instead of a slave is that there is a general expectation that a slave has a relationship with the master that would cause the slave to be notified of changes in the master with a NOTIFY announcement; that is not the case here. It acts a slave because it knows exactly how the master would reply at the time that it retrieve the signed zone, but it is a pseudo-slave because the master has no way of alerting it of changes.

The advantage of a recursive resolver acting as a pseudo-slave is that other resolvers that demand authoritative answers can ask if for those. However, there are few scenarios in which those demanding resolvers exist. The disadvantage of a recursive resolver acting as a pseudo-slave is that there is no way to signal that it is a pseudo-slave and not a real slave. Thus, someone seeing the AA bit set might thing that the resolver is a real slave. This opens the can of worms about trusting the settings of the AA and AD bits in responses.

5. Pros and Cons of this Technique

This is primarily a tracking / discussion section, and the text is kept even looser than in the rest of this doc. These are not ordered.

5.1. Pros

- Junk queries / negative caching - Currently, a significant number of queries to the root servers are "junk" queries. Many of these queries are TLDs that do not (and may never) exist in the root Another significant source of junk is queries where the negative TLD answer did not get cached because the queries are for second-level domains (a negative cache entry for "foo.example" will not cover a subsequent query for "bar.example").

- DoS against the root service - By distributing the contents of the root to many recursive resolvers, the DoS protection for customers of the root servers is significantly increased. A DDoS may still be able to take down some recursive servers, but there is much more root service infrastructure to attack in order to be effective. Of course, there is still a zone distribution system that could be attacked (but it would need to be kept down for a much longer time to cause significant damage, and so far the root has stood up just fine to DDoS.

- Small increase to privacy of requests - This also removes a place where attackers could collect information. Although query name minimization also achieves some of this, it does still leak the TLDs that people behind a resolver are querying for, which may in itself be a concern (for example someone in a homophobic country who is querying for a name in .gay).

5.2. Cons

- Loss of agility in making root zone changes - Currently, if there is an error in the root zone (or someone needs to make an emergency change), a new root zone can be created, and the root server operators can be notified and start serving the new zone

quickly. Of course, this does not invalidate the bad information in (long TTL) cached answers. Notifying every recursive resolver is not feasible. Currently, an "oops" in the root zone will be cached for the TTL of the record by some percentage of servers. Using the technique described above, the information may be cached (by the same percentage of servers) for the refresh time + the TTL of the record

- No central monitoring point - DNS operators lose the ability to monitor the root system. While there is work underway to implement better instrumentation of the root server system, this (potentially) removes the thing to monitor.

- Increased complexity in nameserver software and their operations - Any proposal for recursive servers to copy and serve the root inherently means more code to write and execute. Note that many recursive resolvers are on inexpensive home routers that are rarely (if ever) updated.

- Changes the nature and distribution of traffic hitting the root servers - If all the "good" recursive resolvers deploy root copying, then root servers end up servicing only "bad" recursive resolvers and attack traffic. The roots (could) become what AS112 is for RFC1918.

6. IANA Considerations

This document requires no action from the IANA.

7. Security Considerations

A resolver that uses this mechanism but does not do full DNSSEC validation on the data it uses can obviously cause serious security issues because it can be fooled into giving wrong answers.

[[More?]]

8. Acknowledgements

The editors fully acknowledge that this is not a new concept, and that we have chatted with many people about this. If we have spoken to you and your name is not listed below, let us know.

9. Contributors

The general concept in this document is not new; there have been discussions regarding recursive resolvers copying the root zone for

many years. The fact that the root zone is now signed with DNSSEC makes implementing some of these techniques more feasible.

The following is an unordered list of individuals have contributed text and / or significant discussions to this document.

Steve Crocker - Shinkuro

Jaap Akkerhuis - NLnet Labs

David Conrad - Virtualized, LLC.

Lars-Johan Liman - Netnod

Suzanne Woolf - Individual

Roy Arends - Nominet

Olaf Kolkman - NLnet Labs

Danny McPherson - Verisign

Joe Abley - Dyn

Jim Martin - ISC

Jared Mauch - NTT America

Rob Austien - Dragon Research Labs

Sam Weiler - Parsons

10. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Authors' Addresses

Warren Kumari (editor)
Google
1600 Amphitheatre Parkway
Mountain View, Ca 94043
US

Email: Warren@kumari.net

Paul Hoffman (editor)
VPN Consortium

Email: paul.hoffman@vpnc.org