Network Working Group
Internet-Draft
Obsoletes: 7706 (if approved)
Intended status: Informational
Expires: September 14, 2020

W. Kumari
Google
P. Hoffman
ICANN

March 13, 2020

Running a Root Server Local to a Resolver
draft-ietf-dnsop-7706bis-12

Abstract

Some DNS recursive resolvers have longer-than-desired round-trip times to the closest DNS root server; those resolvers may have difficulty getting responses from the root servers, such as during a network attack. Some DNS recursive resolver operators want to prevent snooping by third parties of requests sent to DNS root servers. In both cases, resolvers can greatly decrease the round-trip time and prevent observation of requests by serving a copy of the full root zone on the same server, such as on a loopback address or in the resolver software. This document shows how to start and maintain such a copy of the root zone that does not cause problems for other users of the DNS, at the cost of adding some operational fragility for the operator.

This document obsoletes RFC 7706.

[This document is being collaborated on in Github at: https://github.com/wkumari/draft-kh-dnsop-7706bis. The most recent version of the document, open issues, and so on should all be available there. The authors gratefully accept pull requests.]

Table of Contents

1. Introduction

DNS recursive resolvers have to provide answers to all queries from their clients, even those for domain names that do not exist. For each queried name that is within a top-level domain (TLD) that is not in the recursive resolver's cache, the resolver must send a query to a root server to get the information for that TLD, or to find out that the TLD does not exist. Research shows that the vast majority

of queries going to the root are for names that do not exist in the root zone.

Many of the queries from recursive resolvers to root servers get answers that are referrals to other servers. Malicious third parties might be able to observe that traffic on the network between the recursive resolver and root servers.

The primary goals of this design are to provide more reliable answers for queries to the root zone during network attacks that affect the root servers, and to prevent queries and responses from being visible on the network. This design will probably have little effect on getting faster responses to stub resolver for good queries on TLDs, because the TTL for most TLDs is usually long-lived (on the order of a day or two) and is thus usually already in the cache of the recursive resolver; the same is true for the TTL for negative answers from the root servers. (Although the primary goal of the design is for serving the root zone, the method can be used for any zone.)

This document describes a method for the operator of a recursive resolver to have a complete root zone locally, and to hide queries for the root zone from outsiders. The basic idea is to create an up-to-date root zone service on the same host as the recursive server, and use that service when the recursive resolver looks up root information. The recursive resolver validates all responses from the root service on the same host, just as it would validate all responses from a remote root server.

This design explicitly only allows the new root zone service to be run on the same server as the recursive resolver, in order to prevent the server from serving authoritative answers to any other system. Specifically, the root service on the local system MUST be configured to only answer queries from resolvers on the same host, and MUST NOT answer queries from any other resolver.

At the time that RFC 7706 [RFC7706] was published, it was considered controversial: there was not consensus on whether this was a "best practice". In fact, many people felt that it is an excessively risky practice because it introduced a new operational piece to local DNS operations where there was not one before. Since then, the DNS operational community has largely shifted to believing that local serving of the root zone for an individual resolver is a reasonable practice. The advantages listed above do not come free: if this new system does not work correctly, users can get bad data, or the entire recursive resolution system might fail in ways that are hard to diagnose.

This design uses authoritative service running on the same machine as the recursive resolver. Common open source recursive resolver software does not need to add new functionality to act as an authoritative server for some zones, but other recursive resolver software might need to be able to talk to an authoritative server running on the same host. Some resolver software supports being both an authoritative server and a resolver but separated by logical "views", allowing a local root to be implemented within a single process; examples of this can be seen in Appendix B.

A different approach to solving some of the problems discussed in this document is described in [RFC8198].

Readers are expected to be familiar with [RFC8499].

1.1. Changes from RFC 7706

RFC 7706 explicitly required that a root server instance be run on the loopback interface of the host running the validating resolver. However, RFC 7706 also had examples of how to set up common software that did not use the loopback interface. This document loosens the restriction on using the loopback interface and in fact allows the use of a local service, not necessarily an authoritative server. However, the document keeps the requirement that only systems running on that single host be able to query that authoritative root server or service.

This document changes the use cases for running a local root service to be more consistent with the reasons operators said they had for using RFC 7706.

Removed the prohibition on distribution of recursive DNS servers including configurations for this design because some already do, and others have expressed an interest in doing so.

Added the idea that a recursive resolver using this design might switch to using the normal (remote) root servers if the local root server fails.

Refreshed the list of where one can get copies of the root zone.

Added examples of other resolvers and updated the existing examples.

1.2. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP

14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Requirements

In order to implement the mechanism described in this document:

- o The system MUST be able to validate every signed record in a zone with DNSSEC [RFC4033].
- o The system MUST have an up-to-date copy of the public part of the Key Signing Key (KSK) [RFC4033] used to sign the DNS root.
- o The system MUST be able to retrieve a copy of the entire root zone (including all DNSSEC-related records).
- o The system MUST be able to run an authoritative service for the root zone on the same host. The authoritative root service MUST only respond to queries from the same host. One way to assure not responding to queries from other hosts is to run an authoritative server for the root that responds only on one of the loopback addresses (that is, an address in the range 127/8 for IPv4 or ::1 in IPv6). Another method is to have the resolver software also act as an authoritative server for the root zone, but only for answering queries from itself.

A corollary of the above list is that authoritative data in the root zone used on the local authoritative server MUST be identical to the same data in the root zone for the DNS. It is possible to change the unsigned data (the glue records) in the copy of the root zone, but such changes could cause problems for the recursive server that accesses the local root zone, and therefore any changes to the glue records SHOULD NOT be made.

3. Operation of the Root Zone on the Local Server

The operation of an authoritative server for the root in the system described here can be done separately from the operation of the recursive resolver, or it might be part of the configuration of the recursive resolver system.

The steps to set up the root zone are:

1. Retrieve a copy of the root zone. (See Appendix A for some current locations of sources.)

2. Start the authoritative service for the root zone in a manner that prevents any system other than a recursive resolver on the same host from accessing it.

The contents of the root zone MUST be refreshed using the timers from the SOA record in the root zone, as described in [RFC1035]. This inherently means that the contents of the local root zone will likely be a little behind those of the global root servers because those servers are updated when triggered by NOTIFY messages.

There is a risk that a system using a local authoritative server for the root zone cannot refresh the contents of the root zone before the expire time in the SOA. A system using a local authoritative server for the root zone MUST NOT serve stale data for the root zone. To mitigate the risk that stale data is served, the local root server MUST immediately switch to using non-local root servers when it detects that it would be serving state data.

In a resolver that is using an internal service for the root zone, if the contents of the root zone cannot be refreshed before the expire time in the SOA, the resolver MUST immediately switch to using non-local root servers.

In the event that refreshing the contents of the root zone fails, the results can be disastrous. For example, sometimes all the NS records for a TLD are changed in a short period of time (such as 2 days); if the refreshing of the local root zone is broken during that time, the recursive resolver will have bad data for the entire TLD zone.

An administrator using the procedure in this document SHOULD have an automated method to check that the contents of the local root zone are being refreshed; this might be part of the resolver software. One way to do this is to have a separate process that periodically checks the SOA of the local root zone and makes sure that it is changing. At the time that this document is published, the SOA for the root zone is the digital representation of the current date with a two-digit counter appended, and the SOA is changed every day even if the contents of the root zone are unchanged. For example, the SOA of the root zone on January 2, 2019 was 2019010201. A process can use this fact to create a check for the contents of the local root zone (using a program not specified in this document).

4. Security Considerations

A system that does not follow the DNSSEC-related requirements given in Section 2 can be fooled into giving bad responses in the same way as any recursive resolver that does not do DNSSEC validation on responses from a remote root server. Anyone deploying the method

described in this document should be familiar with the operational benefits and costs of deploying DNSSEC [RFC4033].

As stated in Section 1, this design explicitly requires the local copy of the root zone information to be available only from resolvers on that host. This has the security property of limiting damage to clients of any local resolver that might try to rely on an altered copy of the root.

5. IANA Considerations

This document has no actions for IANA.

6. References

6.1. Normative References

[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>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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>.

[RFC7706] Kumari, W. and P. Hoffman, "Decreasing Access Time to Root Servers by Running One on Loopback", RFC 7706, DOI 10.17487/RFC7706, November 2015, <https://www.rfc-editor.org/info/rfc7706>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

6.2. Informative References

[Manning2013]
Manning, W., "Client Based Naming", 2013, <http://www.sfc.wide.ad.jp/dissertation/bill_e.html>.

[RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <https://www.rfc-editor.org/info/rfc5936>.

[RFC8198] Fujiwara, K., Kato, A., and W. Kumari, "Aggressive Use of DNSSEC-Validated Cache", RFC 8198, DOI 10.17487/RFC8198, July 2017, <https://www.rfc-editor.org/info/rfc8198>.

Appendix A. Current Sources of the Root Zone

The root zone can be retrieved from anywhere as long as it comes with all the DNSSEC records needed for validation. Currently, one can get the root zone from ICANN by zone transfer (AXFR) [RFC5936] over TCP from DNS servers at xfr.lax.dns.icann.org and xfr.cjr.dns.icann.org. The root zone file can be obtained using methods described at <https://www.iana.org/domains/root/files>.

Currently, the root can also be retrieved by AXFR over TCP from the following root server operators:

- b.root-servers.net
- c.root-servers.net
- d.root-servers.net
- f.root-servers.net
- g.root-servers.net
- k.root-servers.net

It is crucial to note that none of the above services are guaranteed to be available. It is possible that ICANN or some of the root server operators will turn off the AXFR capability on the servers listed above. Using AXFR over TCP to addresses that are likely to be anycast (as the ones above are) may conceivably have transfer problems due to anycast, but current practice shows that to be unlikely.

A.1. Root Zone Services

At the time that this document is published, there is one root zone service that is active, and one that has been announced as in the planning stages. This section describes all known active services.

LocalRoot (<https://localroot.isi.edu/>) is an experimental service that embodies many of the ideas in this document. It distributes the root zone by AXFR, and also offers DNS NOTIFY messages when the LocalRoot system sees that the root zone has changed.

Appendix B. Example Configurations of Common Implementations

This section shows fragments of configurations for some popular recursive server software that is believed to correctly implement the requirements given in this document. The examples have been updated since the publication of RFC 7706.

The IPv4 and IPv6 addresses in this section were checked in March 2020 by testing for AXFR over TCP from each address for the known single-letter names in the root-servers.net zone.

B.1. Example Configuration: BIND 9.12

BIND 9.12 acts both as a recursive resolver and an authoritative server. Because of this, there is "fate-sharing" between the two servers in the following configuration. That is, if the root server dies, it is likely that all of BIND is dead.

Note that a future version of BIND will support a much more robust method for creating a local mirror of the root or other zones; see Appendix B.3.

Using this configuration, queries for information in the root zone are returned with the AA bit not set.

When slaving a zone, BIND 9.12 will treat zone data differently if the zone is slaved into a separate view (or a separate instance of the software) versus slaved into the same view or instance that is also performing the recursion.

Validation: When using separate views or separate instances, the DS records in the slaved zone will be validated as the zone data is accessed by the recursive server. When using the same view, this validation does not occur for the slaved zone.

Caching: When using separate views or instances, the recursive server will cache all of the queries for the slaved zone, just as

it would using the traditional "root hints" method. Thus, as the zone in the other view or instance is refreshed or updated, changed information will not appear in the recursive server until the TTL of the old record times out. Currently, the TTL for DS and delegation NS records is two days. When using the same view, all zone data in the recursive server will be updated as soon as it receives its copy of the zone.

```
view root {
    match-destinations { 127.12.12.12; };
    zone "." {
        type slave;
        file "rootzone.db";
        notify no;
        masters {
            199.9.14.201;         # b.root-servers.net
            192.33.4.12;          # c.root-servers.net
            199.7.91.13;          # d.root-servers.net
            192.5.5.241;          # f.root-servers.net
            192.112.36.4;         # g.root-servers.net
            193.0.14.129;         # k.root-servers.net
            192.0.47.132;         # xfr.cjr.dns.icann.org
            192.0.32.132;         # xfr.lax.dns.icann.org
            2001:500:200::b;      # b.root-servers.net
            2001:500:2::c;        # c.root-servers.net
            2001:500:2d::d;       # d.root-servers.net
            2001:500:2f::f;       # f.root-servers.net
            2001:500:12::d0d;     # g.root-servers.net
            2001:7fd::1;          # k.root-servers.net
            2620:0:2830:202::132; # xfr.cjr.dns.icann.org
            2620:0:2d0:202::132;  # xfr.lax.dns.icann.org
        };
    };
};

view recursive {
    dnssec-validation auto;
    allow-recursion { any; };
    recursion yes;
    zone "." {
        type static-stub;
        server-addresses { 127.12.12.12; };
    };
};
```

B.2. Example Configuration: Unbound 1.8

Similar to BIND, Unbound starting with version 1.8 can act both as a recursive resolver and an authoritative server.

```
auth-zone:
    name: "."
    master: 199.9.14.201         # b.root-servers.net
    master: 192.33.4.12          # c.root-servers.net
    master: 199.7.91.13          # d.root-servers.net
    master: 192.5.5.241          # f.root-servers.net
    master: 192.112.36.4         # g.root-servers.net
    master: 193.0.14.129         # k.root-servers.net
    master: 192.0.47.132         # xfr.cjr.dns.icann.org
    master: 192.0.32.132         # xfr.lax.dns.icann.org
    master: 2001:500:200::b      # b.root-servers.net
    master: 2001:500:2::c        # c.root-servers.net
    master: 2001:500:2d::d       # d.root-servers.net
    master: 2001:500:2f::f       # f.root-servers.net
    master: 2001:500:12::d0d     # g.root-servers.net
    master: 2001:7fd::1          # k.root-servers.net
    master: 2620:0:2830:202::132 # xfr.cjr.dns.icann.org
    master: 2620:0:2d0:202::132  # xfr.lax.dns.icann.org
    fallback-enabled: yes
    for-downstream: no
    for-upstream: yes
```

B.3. Example Configuration: BIND 9.14

BIND 9.14 can set up a local mirror of the root zone with a small configuration option:

```
zone "." {
    type mirror;
};
```

The simple "type mirror" configuration for the root zone works for the root zone because a default list of primary servers for the IANA root zone is built into BIND 9.14. In order to set up mirroring of any other zone, an explicit list of primary servers needs to be provided.

See the documentation for BIND 9.14 for more detail about how to use this simplified configuration.

B.4. Example Configuration: Unbound 1.9

Recent versions of Unbound have a "auth-zone" feature that allows local mirroring of the root zone. Configuration looks like:

```
auth-zone:
    name: "."
    master: "b.root-servers.net"
    master: "c.root-servers.net"
    master: "d.root-servers.net"
    master: "f.root-servers.net"
    master: "g.root-servers.net"
    master: "k.root-servers.net"
        fallback-enabled: yes
    for-downstream: no
    for-upstream: yes
    zonefile: "root.zone"
```

B.5. Example Configuration: Knot Resolver

Knot Resolver uses its "prefill" module to load the root zone information. This is described at <https://knot-resolver.readthedocs.io/en/v5.0.1/modules-rfc7706.html>.

B.6. Example Configuration: Microsoft Windows Server 2012

Windows Server 2012 contains a DNS server in the "DNS Manager" component. When activated, that component acts as a recursive server. DNS Manager can also act as an authoritative server.

Using this configuration, queries for information in the root zone are returned with the AA bit set.

The steps to configure DNS Manager to implement the requirements in this document are:

1. Launch the DNS Manager GUI. This can be done from the command line ("dnsmgmt.msc") or from the Service Manager (the "DNS" command in the "Tools" menu).

2. In the hierarchy under the server on which the service is running, right-click on the "Forward Lookup Zones", and select "New Zone". This brings up a succession of dialog boxes.

3. In the "Zone Type" dialog box, select "Secondary zone".

4. In the "Zone Name" dialog box, enter ".".

5. In the "Master DNS Servers" dialog box, enter "b.root-servers.net". The system validates that it can do a zone transfer from that server. (After this configuration is completed, the DNS Manager will attempt to transfer from all of the root zone servers.)

6. In the "Completing the New Zone Wizard" dialog box, click "Finish".

7. Verify that the DNS Manager is acting as a recursive resolver. Right-click on the server name in the hierarchy, choosing the "Advanced" tab in the dialog box. See that "Disable recursion (also disables forwarders)" is not selected, and that "Enable DNSSEC validation for remote responses" is selected.

Acknowledgements

The authors fully acknowledge that running a copy of the root zone on the loopback address is not a new concept, and that we have chatted with many people about that idea over time. For example, Bill Manning described a similar solution to the problems in his doctoral dissertation in 2013 [Manning2013].

Evan Hunt contributed greatly to the logic in the requirements. Other significant contributors include Wouter Wijngaards, Tony Hain, Doug Barton, Greg Lindsay, and Akira Kato. The authors also received many offline comments about making the document clear that this is just a description of a way to operate a root zone on the same host, and not a recommendation to do so.

People who contributed to this update to RFC 7706 include: Florian Obser, nusenu, Wouter Wijngaards, Mukund Sivaraman, Bob Harold, and Leo Vegoda.

Authors' Addresses

Warren Kumari
Google

Email: Warren@kumari.net

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org