Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: July 16, 2016

P. Koch
DENIC eG
M. Larson
Dyn, Inc.
P. Hoffman
ICANN

January 13, 2016

Initializing a DNS Resolver with Priming Queries
draft-ietf-dnsop-resolver-priming-06

Abstract

This document describes the queries a DNS resolver can emit to initialize its cache. The result is that the resolver gets both a current NS RRSet for the root zone and the necessary address information for reaching the root servers.

1. Introduction

Recursive DNS resolvers need a starting point to resolve queries. [RFC1034] describes a common scenario for recursive resolvers: they begin with an empty cache and some configuration for finding the names and addresses of the DNS root servers. [RFC1034] describes that configuration as a list of servers that will authoritative answers to queries about the root. This has become a common implementation choice for recursive resolvers, and is the topic of this document.

This document describes the steps needed for this common implementation choice. Note that this is not the only way to start a recursive name server with an empty cache, but it is the only one described in [RFC1034]. Some implementers have chosen other directions, some of which work well and others of which fail (sometimes disastrously) under different conditions.

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].

This document only deals with recursive name servers (recursive resolvers, resolvers) for the IN class.

2. Description of Priming

As described in this document, priming is the act of finding the list of root servers from a configuration that lists some or all of the purported IP addresses of some or all of those root servers. A recursive resolver starts with no information about the root servers, and ends up with a list of their names and their addresses.

Priming is described in Sections 5.3.2 and 5.3.3 of [RFC1034]. The scenario used in that description, that of a recursive server that is also authoritative, is no longer as common.

Currently, it is quite common for the configured list of IP addresses for the root server to be mostly complete and correct. Note that this list (at least initially) comes from the vendor or distributor of the recursive server software.

The list of root server operators and the domain name associated with each one has been stable since 1997. However, there are address changes for the NS records for those root server operators, both for

IPv4 and IPv6 addresses. However, research shows that after those addresses change, some resolvers never get the new addresses. Therefore, it is important that resolvers be able to cope with change, even without relying upon configuration updates to be applied by their operator. This is the main reason that one needs to do priming instead of just going from a configured list to get a full and accurate list of root servers.

3. Priming Queries

A priming query is a DNS query that has a QNAME of "." and a QTYPE of NS, and is sent to one of the addresses in the configuration for the recursive resolver. The priming query MAY be sent over UDP or TCP. If the query is sent over UDP, the source port SHOULD be randomly selected (see [RFC5452]). The RD bit MAY be set to 0 or 1, although the meaning of it being set to 1 is undefined for priming queries.

The recursive resolver SHOULD use EDNS0 [RFC6891] for priming queries and SHOULD announce and handle a reassembly size of at least 1024 octets [RFC3226]. Doing so allows responses that cover the size of a full priming response (see Section 4.2).

3.1. Repeating Priming Queries

The recursive resolver SHOULD send a priming query only when it is needed. This would be when the resolver starts with an empty cache, and when one or more of the NS records for the root servers has expired. The recursive resolver SHOULD expire the NS records of the root servers according to the TTL values given in the priming response.

If a priming query does not get a response within 2 seconds, the recursive resolver SHOULD retry with a different target address from the configuration.

3.2. Target Selection

In order to spread the load across all the root server operators, the recursive resolver SHOULD select the target for a priming query randomly from the list of addresses. The recursive resolver might choose either IPv4 and IPv6 addresses based on its knowledge of whether the server on which it is running has adequate transit on either type of address.

Note that this recommended method is not the only way to choose from the list in a recursive resolver's configuration. Two other other common methods include picking the first from the list, and remembering which address in the list gave the fastest response

earlier and using that one. There are probably other methods in use today. However, the random method listed above is the one that is recommended for priming.

3.3. DNSSEC with Priming Queries

The resolver MAY set the DNSSEC OK [RFC4033] bit. At the time this document is being published, there is little use to performing DNSSEC validation on the priming query because the "root-servers.net" zone is not signed, and so a man-in-the-middle attack on the priming query can result in malicious data in the responses. However, if the "root-servers.net" zone is later signed, or if the root server operators choose a different zone to identify themselves and that zone is signed, having DNSSEC validation for the priming queries might be valuable.

4. Priming Responses

A priming query is a normal DNS query. Thus, a root name server cannot distinguish a priming query from any other query for the root NS RRSet. Thus, the root server's response will also be a normal DNS response.

4.1. Expected Properties of the Priming Response

The priming response is expected to have an RCODE of NOERROR, and to have the AA bit set. Also, there should be an NS RRSet in the Answer section (because the NS RRSet originates from the root zone), an empty Authority section (because the NS RRSet already appears in the answer section) and an Additional section with A and/or AAAA RRSets for the root name servers pointed at by the NS RRSet.

Resolver software SHOULD treat the response to the priming query as a normal DNS response, just as it would use any other data fed to its cache. Resolver software SHOULD NOT expect exactly 13 NS RRs.

4.2. Completeness of the Response

There are currently 13 root servers. Of those 13, all have one IPv4 address, and 11 have an IPv6 address. The combined size of all the A and AAAA RRSets is (13 * 16) + (11 * 32), or 560 bytes. Not even counting the NS RRSet, this value exceeds the original 512 octet payload limit from [RFC1035].

For an EDNS response, a resolver SHOULD consider the address information found in the Additional section complete for any particular server that appears at all. Said another way: in an EDNS

response, if the additional section only has an A RRSet for a server, the resolver SHOULD assume that no AAAA RRSet exists.

It is important to note that if the recursive resolver did not announce a reassembly size larger than 512 octets, this assumption is invalid. Re-issuing of the priming query does not help with those root name servers that respond with a fixed order of addresses in the additional section. Instead, the recursive resolver needs to issue direct queries for A and AAAA RRSets for the remaining names. Currently, these RRSets would be authoritatively available from the root name servers.

5. Security Considerations

Spoofing a response to a priming query can be used to redirect all of the queries originating from a victim recursive resolver to one or more servers for the attacker. Until the responses to priming queries are protected with DNSSEC, there is no definitive way to prevent such redirection.

6. IANA Considerations

None.

7. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[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>.

[RFC3226] Gudmundsson, O., "DNSSEC and IPv6 A6 aware server/resolver message size requirements", RFC 3226, DOI 10.17487/RFC3226, December 2001, <http://www.rfc-editor.org/info/rfc3226>.

[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, <http://www.rfc-editor.org/info/rfc4033>.

[RFC5452] Hubert, A. and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers", RFC 5452, DOI 10.17487/RFC5452, January 2009, <http://www.rfc-editor.org/info/rfc5452>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <http://www.rfc-editor.org/info/rfc6891>.

Appendix A. Acknowledgements

This document is the product of the DNSOP WG and benefitted from the reviews done there.

Authors' Addresses

Peter Koch
DENIC eG
Kaiserstrasse 75-77
Frankfurt 60329
DE

Phone: +49 69 27235 0
Email: pk@DENIC.DE

Matt Larson
Dyn, Inc.
150 Dow St
Manchester, NH 03101
USA

Email: mlarson@dyn.com

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org