Network Working Group
Internet-Draft
Expires: January 19, 2006

R. Austein
ISC

July 18, 2005

EDNS NSID Extension
draft-austein-dnsext-nsid-02

Status of this Memo

Copyright Notice

Abstract

With the increased use of DNS anycast, load balancing, and other mechanisms allowing more than one DNS name server to share a single IP address, it is sometimes difficult to tell which of a pool of name servers has answered a particular query. While existing ad-hoc mechanism allow an operator to send follow-up queries when it is necessary to debug such a configuration, the only completely reliable way to obtain the identity of the name server which actually responded is to have the name server include this information in the response itself. This note proposes a protocol extension to support

this functionality.

Table of Contents

1. Introduction

With the increased use of DNS anycast, load balancing, and other mechanisms allowing more than one DNS name server to share a single IP address, it is sometimes difficult to tell which of a pool of name servers has answered a particular query.

Existing ad-hoc mechanisms such as those described in [I-D.ietf-dnsop-serverid] allow an operator to send follow-up queries when it is necessary to debug such a configuration, but there are situations in which this is not a totally satisfactory solution, since anycast routing may have changed, or the server pool in question may be behind some kind of extremely dynamic load balancing hardware. Thus, while these ad-hoc mechanisms are certainly better than nothing (and have the advantage of already being deployed), a better solution seems desirable.

Given that a DNS query is an idempotent operation with no retained state, it would appear that the only completely reliable way to obtain the identity of the name server which actually responded to a particular query is to have that name server include identifying information in the response itself. This note proposes a protocol enhancement to achieve this.

2. Proposed Mechanism

This note proposes using an EDNS [RFC2671] flag bit to signal the resolver's desire for information identifying the name server, and an EDNS option to hold the name server's response (should it choose to honor the resolver's request).

2.1 The SI Flag

A resolver signals its desire for information identifying the server by setting the SI (Send Identification) flag in the extended flags field of the OPT pseudo-RR.

The value of the SI flag is [TBD].

The semantics of the SI flag are not transitive. That is: the SI flag is a request that the name server which receives the query identify itself; in a so-called forwarding setup, the first hop name server is the one that should identify itself. If the resolver side of a forwarding name server wishes to receive identifying information, it is free to set the SI flag in its own queries, but that is a separate matter.

A name server which understands the SI flag should echo its value back in the response message, regardless of whether the name server chose to honor the request.

2.2 The NSID Option

A name server which understands the SI flag and chooses to honor it responds by including identifying information in a NSID option in an EDNS OPT pseudo-RR in the response message.

The OPTION-CODE for the NSID option is [TBD].

The OPTION-DATA for the NSID option is an opaque byte string the semantics of which are deliberately left outside the protocol. See Section 3 for discussion.

The NSID option is not transitive. A name server must not send an NSID option back to a resolver which did not request it. In particular, while a forwarder may choose to set the SI bit when forwarding a query, this has no effect on the setting of the SI bit or the presence or absence of the NSID option in the forwarder's response to the original client.

3. What Should the NSID Payload Be?

The syntax and semantics of the content of the NSID option is deliberately left outside the scope of this specification. This describe some of the kinds of data that server administrators might choose to provide as the content of the NSID option, and explains the reasoning (such as it is) behind choosing a simple opaque byte string.

There are several possibilities for the payload of the NSID option.

- It could be the "real" name of the specific name server within the name server pool.
- It could be the "real" IP address (IPv4 or IPv6) of the name server within the name server pool.
- It could be some sort of pseudo-random number generated in a predictable fashion somehow using the server's IP address or name as a seed value.
- It could be some sort of probabilisticly unique identifier initially derived from some sort of random number generator then preserved across reboots of the name server.
- It could be some sort of dynamicly generated identifier so that only the name server operator could even tell whether or not any two queries had been answered by the same server.
- It could be a blob of signed data, with a corresponding key which might (or might not) be available via DNS lookups.
- It could be a blob of encrypted data, the key for which presumably would be restricted to parties with a need to know (in the opinion of the server operator).
- It could be an arbitrary string of octets chosen at the discretion of the name server operator.

Each of these options has advantages and disadvantages.

- Using the "real" name is simple, but assumes that the name server has a "real" name, which it may not.
- Using the "real" address is also simple, and the name server almost certainly does have at least one non-anycast IP address for maintenance operations, but assumes that the operator of the name server is willing to divulge its non-anycast address, which might

not be the case.

- Given that one common reason for using anycast DNS techniques is an attempt to harden a critical name server against denial of service attacks, some name server operators are likely to want an identifier other than the "real" name or "real" address of the name server instance.

- Using a hash or pseudo-random number can provide a fixed length value that the resolver can use to tell two name servers apart without necessarily being able to tell where either one of them "really" is, but makes debugging more difficult if one happens to be in a friendly open environment. Furthermore, a nonce may not add much value, since a hash based on an IPv4 address still only involves a 32-bit search space, and DNS names used for servers that operators might have to debug at 4am tend not to be very random at all.

- Probabilisticly unique identifiers have similar properties to hashed identifiers, but (given a sufficiently good random number generator) are immune to the search space issues. However, the strength of this approach is also its weakness: there is no algorithmic transformation by which even the server operator can associate name server instances with identifiers while debugging, which might be annoying. This approach also requires the name server instance to preserve the probabilisticly unique identifier across reboots, but this does not appear to be a serious restriction, since authoritative nameservers almost always have nonvolatile storage (such as a disk drive) in any case, and in rare cases where a name server does not have any way to store such an identifier, nothing terrible will happen if the name server just generates a new identifier every time it reboots.

- Using an arbitrary octet string gives name server operators yet another thing to configure, or mis-configure, or forget to configure. Having all the nodes in an anycast name server constellation identify themselves as "My Name Server" would not be particularly useful.

Given all of the issues listed above, the best approach might be:

- Define the NSID payload to be an opaque byte string, as specified in Section 2.2.

- Operators for whom divulging the unicast address is an issue could use the raw binary representation of a probabilisticly unique random number. This should probably be the default implementation behavior.

- Operators for whom divulging the unicast address is not an issue could just use the raw binary representation of a unicast address for simplicity. This would only be done via an explicit configuration choice by the operator.

- Operators who really need or want the ability to set the NSID payload to an arbitrary value could do so, but this would only be done via an explicit configuration choice by the operator.

This approach appears to provide enough information for useful debugging without unintentionally leaking the maintenance addresses of anycast name servers to nogoodniks, while also allowing name server operators who do not find such leakage threatening to provide more information at their own discretion.

4. Should Recursive Name Servers Respond to SI?

Most of the discussion of name server identification to date has focused on identifying authoritative name servers, since the best known cases of anycast name servers are a subset of the name servers for the root zone. However, given that anycast DNS techniques are equally applicable to recursive name servers as well as authoritative name servers, it may be useful for the name server side of a recursive name server to support this mechanism as well. The semantics proposed for the SI bit in Section 2.1 are intended to support this model.

5. IANA Considerations

This mechanism requires allocation of one EDNS flag bit for the SI flag (Section 2.1).

This mechanism requires allocation of one ENDS option code for the NSID option (Section 2.2).

6. Security Considerations

This document describes a channel signaling mechanism, intended primarily for debugging. Channel signaling mechanisms are outside the scope of DNSSEC per se. Thus, applications that require integrity protection for the data being signaled will need to use a channel security mechanism such as TSIG [RFC2845].

Section 3 discusses a number of different kinds of information that a name server operator might choose to provide as the value of the NSID option. Some of these kinds of information are security sensitive in some environments. This specification deliberately leaves the syntax and semantics of the NSID option content up to the implementation and the name server operator.

7. Acknowledgements

Joe Abley, Harald Alvestrand, Roy Arends, Steve Bellovin, Randy Bush, David Conrad, Johan Ihren, Mike Patton, Paul Vixie, Sam Weiler, Suzanne Woolf, and the law firm of Dewey, Chetham, and Howe.

8. References

8.1 Normative References

[RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, May 2000.

8.2 Informative References

[I-D.ietf-dnsop-serverid] Conrad, D., "Identifying an Authoritative Name Server", draft-ietf-dnsop-serverid-04 (work in progress), March 2005.

Author's Address

Rob Austein
ISC
950 Charter Street
Redwood City, CA 94063
USA

Email: sra@isc.org

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.