Network Working Group
Internet-Draft
Expires: March 13, 2006

R. Austein
ISC

September 9, 2005

DNS Name Server Identifier Option (NSID)
draft-ietf-dnsext-nsid-00

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 responded is to have the name server include this information in the response itself. This note defines 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 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 responded to a particular query is to have that name server include identifying information in the response itself. This note defines a protocol enhancement to achieve this.

1.1 Reserved Words

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. Protocol

This note uses 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, if any.

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. If the name server side of a recursive name server receives the SI bit, the client is asking the recursive name server to identify itself; if the resolver side of the recursive 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 3.1 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 recursive name server may choose to set the SI bit when sending a query, this has no effect on the setting of the SI bit or the presence or absence of the NSID option in the recursive name server's response to the original client.

As stated in Section 2.1, this mechanism is not restricted to authoritative name servers; the semantics are intended to be equally applicable to recursive name servers.

2.3 Presentation Format

User interfaces MUST read and write the content of the NSID option as a sequence of hexadecimal digits, two digits per payload octet.

The NSID payload is binary data. Any comparison between NSID payloads MUST be a comparison of the raw binary data. Copy operations MUST NOT assume that the raw NSID payload is null-terminated. Any resemblance between raw NSID payload data and any form of text is purely a convenience, and does not change the underlying nature of the payload data.

See Section 3.3 for discussion.

3. Discussion

This section discusses certain aspects of the protocol and explains considerations that led to the chosen design.

3.1 The NSID Payload

The syntax and semantics of the content of the NSID option is deliberately left outside the scope of this specification. This section 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 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 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 could 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 the name server may not have a "real" name.
- 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 the operator of the name server may not be willing to divulge its non-anycast address.
- 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, hashing might 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.
- 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 some form of nonvolatile storage in any case, and in the rare case of a name server that 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, there does not appear to be a single solution that will meet all needs. Section 2.2 therefore defines the NSID payload to be an opaque byte string and leaves the choice up to the implementor and name server operator. The following guidelines may be useful to implementors and server operators:

- 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 should 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 should 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.

3.2 SI and NSID Are Not Transitive

As specified in Section 2.1 and Section 2.2, the SI flag and NSID option are not transitive. This is strictly a hop-by-hop mechanism.

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 also applicable to recursive name servers, the mechanism may also be useful with recursive name servers. The hop-by-hop semantics support this.

While there might be some utility in having a transitive variant of this mechanism (so that, for example, a stub resolver could ask a recursive server to tell it which authoritative name server provided a particular answer to the recursive name server), the semantics of such a variant would be more complicated, and are left for future work.

3.3 User Interface Issues

Given the range of possible payload contents described in Section 3.1, it is not possible to define a single presentation format for the NSID payload that is efficient, convenient, unambiguous, and aesthetically pleasing. In particular, while it is tempting to use a presentation format that uses some form of textual strings, attempting to support this would significantly complicate what's intended to be a very simple debugging mechanism.

In some cases the content of the NSID payload may binary data only be meaningful to the name server operator, and may not be meaningful to the user or application, but the user or application must be able to capture the entire content anyway in order for it to be useful. Thus, the presentation format must support arbitrary binary data.

In cases where the name server operator derives the NSID payload from textual data, a textual form such as US-ASCII or UTF-8 strings might at first glance seem easier for a user to deal with. There are, however, a number of complex issues involving internationalized text which, if fully addressed here, would require a set of rules significantly longer than the rest of this specification. See [RFC2277] for an overview of some of these issues.

It is much more important for the NSID payload data to be passed unambiguously from server administrator to user than it is for the payload data data to be pretty while in transit. In particular, it's critical that it be straightforward for a user to cut and paste an

exact copy of the NSID payload output by a debugging tool into other

formats such as email messages or web forms without distortion. Hexadecimal strings, while ugly, are also robust.

4. 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).

5. Security Considerations

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

Section 3.1 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.

6. Acknowledgements

Joe Abley, Harald Alvestrand, Mark Andrews, Roy Arends, Steve Bellovin, Randy Bush, David Conrad, Johan Ihren, Daniel Karrenberg, Mike Patton, Paul Vixie, Sam Weiler, and Suzanne Woolf. Apologies to anyone inadvertently omitted from the above list.

7. References

7.1 Normative References

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

[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.

7.2 Informative References

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

Author's Address

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

Email: sra@isc.org

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement

Acknowledgment

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