Network Working Group
Request for Comments: 5001
Category: Standards Track

R. Austein
ISC

August 2007

DNS Name Server Identifier (NSID) Option

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

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 mechanisms 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 that 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 that 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] option to signal the resolver’s desire for information identifying the name server and to hold the name server’s response, if any.

2.1. Resolver Behavior

A resolver signals its desire for information identifying a name server by sending an empty NSID option (Section 2.3) in an EDNS OPT pseudo-RR in the query message.

The resolver MUST NOT include any NSID payload data in the query message.

The semantics of an NSID request are not transitive. That is: the presence of an NSID option in a query is a request that the name server which receives the query identify itself. If the name server side of a recursive name server receives an NSID request, 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 add NSID requests in its own queries, but that is a separate matter.

2.2. Name Server Behavior

A name server that understands the NSID option and chooses to honor a particular NSID request responds by including identifying information in a NSID option (Section 2.3) in an EDNS OPT pseudo-RR in the response message.

The name server MUST ignore any NSID payload data that might be present in the query message.

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 add an NSID option when sending a query, this has no effect on 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. The NSID Option

The OPTION-CODE for the NSID option is 3.

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.

2.4. Presentation Format

User interfaces MUST read and write the contents 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 are deliberately left outside the scope of this specification.

Choosing the NSID content is a prerogative of the server administrator. The server administrator might choose to encode the NSID content in such a way that the server operator (or clients authorized by the server operator) can decode the NSID content to obtain more information than other clients can. Alternatively, the server operator might choose unencoded NSID content that is equally meaningful to any client.

This section describes 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 specifying a simple opaque byte string in Section 2.3.

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 probabilistically 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 dynamically 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.

- Probabilistically unique identifiers have properties similar 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 probabilistically unique identifier across reboots, but this does not appear to be a serious restriction, since authoritative nameservers almost always have some form of non-volatile storage. 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 generates a new identifier every time it reboots.

- Using an arbitrary octet string gives name server operators yet another setting 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.

- A signed blob is not particularly useful as an NSID payload unless the signed data is dynamic and includes some kind of replay protection, such as a timestamp or some kind of data identifying the requestor. Signed blobs that meet these criteria could conceivably be useful in some situations but would require detailed security analysis beyond the scope of this document.

- A static encrypted blob would not be particularly useful, as it would be subject to replay attacks and would, in effect, just be a random number to any party that does not possess the decryption key. Dynamic encrypted blobs could conceivably be useful in some situations but, as with signed blobs, dynamic encrypted blobs would require detailed security analysis beyond the scope of this document.

Given all of the issues listed above, there does not appear to be a single solution that will meet all needs. Section 2.3 therefore defines the NSID payload to be an opaque byte string and leaves the choice of payload 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 probabilistically 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. NSID Is Not Transitive

As specified in Section 2.1 and Section 2.2, the NSID option is 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 be binary data meaningful only 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 and back again than it is for the payload 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.

3.4. Truncation

In some cases, adding the NSID option to a response message may trigger message truncation. This specification does not change the rules for DNS message truncation in any way, but implementors will need to pay attention to this issue.

Including the NSID option in a response is always optional, so this specification never requires name servers to truncate response messages.

By definition, a resolver that requests NSID responses also supports EDNS, so a resolver that requests NSID responses can also use the "sender’s UDP payload size" field of the OPT pseudo-RR to signal a receive buffer size large enough to make truncation unlikely.

4. IANA Considerations

IANA has allocated EDNS option code 3 for the NSID option (Section 2.3).

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.

Two of the possible kinds of payload data discussed in Section 3.1 involve a digital signature and encryption, respectively. While this specification discusses some of the pitfalls that might lurk for careless users of these kinds of payload data, full analysis of the issues that would be involved in these kinds of payload data would require knowledge of the content to be signed or encrypted, algorithms to be used, and so forth, which is beyond the scope of this document. Implementors should seek competent advice before attempting to use these kinds of NSID payloads.

6. Acknowledgements

Thanks to: Joe Abley, Harald Alvestrand, Dean Anderson, Mark Andrews, Roy Arends, Steve Bellovin, Alex Bligh, Randy Bush, David Conrad, John Dickinson, Alfred Hoenes, Johan Ihren, Daniel Karrenberg, Peter Koch, William Leibzon, Ed Lewis, Thomas Narten, Mike Patton, Geoffrey Sisson, Andrew Sullivan, Mike StJohns, Tom Taylor, Paul Vixie, Sam Weiler, and Suzanne Woolf, none of whom are responsible for what the author did with their comments and suggestions. 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", RFC 2119, BCP 14, March 1997.

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

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, 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", RFC 2277, BCP 18, January 1998.

Author's Address

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

EMail: sra@isc.org

Full Copyright Statement

Intellectual Property

Acknowledgement

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