Network Working Group
Request for Comments: 4697
BCP: 123
Category: Best Current Practice

M. Larson
P. Barber
VeriSign, Inc.

October 2006

Observed DNS Resolution Misbehavior

Status of This Memo

Copyright Notice

Abstract

This memo describes DNS iterative resolver behavior that results in a significant query volume sent to the root and top-level domain (TLD) name servers. We offer implementation advice to iterative resolver developers to alleviate these unnecessary queries. The recommendations made in this document are a direct byproduct of observation and analysis of abnormal query traffic patterns seen at two of the thirteen root name servers and all thirteen com/net TLD name servers.

Table of Contents

1. Introduction

Observation of query traffic received by two root name servers and the thirteen com/net Top-Level Domain (TLD) name servers has revealed that a large proportion of the total traffic often consists of "requeries". A requery is the same question (<QNAME, QTYPE, QCLASS>) asked repeatedly at an unexpectedly high rate. We have observed requeries from both a single IP address and multiple IP addresses (i.e., the same query received simultaneously from multiple IP addresses).

By analyzing requery events, we have found that the cause of the duplicate traffic is almost always a deficient iterative resolver, stub resolver, or application implementation combined with an operational anomaly. The implementation deficiencies we have identified to date include well-intentioned recovery attempts gone awry, insufficient caching of failures, early abort when multiple levels of indirection must be followed, and aggressive retry by stub resolvers or applications. Anomalies that we have seen trigger requery events include lame delegations, unusual glue records, and anything that makes all authoritative name servers for a zone unreachable (Denial of Service (DoS) attacks, crashes, maintenance, routing failures, congestion, etc.).

In the following sections, we provide a detailed explanation of the observed behavior and recommend changes that will reduce the requery rate. None of the changes recommended affects the core DNS protocol specification; instead, this document consists of guidelines to implementors of iterative resolvers.

1.1. A Note about Terminology in This Memo

To recast an old saying about standards, the nice thing about DNS terms is that there are so many of them to choose from. Writing or talking about DNS can be difficult and can cause confusion resulting from a lack of agreed-upon terms for its various components. Further complicating matters are implementations that combine multiple roles into one piece of software, which makes naming the result problematic. An example is the entity that accepts recursive queries, issues iterative queries as necessary to resolve the initial recursive query, caches responses it receives, and which is also able to answer questions about certain zones authoritatively. This entity is an iterative resolver combined with an authoritative name server and is often called a "recursive name server" or a "caching name server".

This memo is concerned principally with the behavior of iterative resolvers, which are typically found as part of a recursive name server. This memo uses the more precise term "iterative resolver", because the focus is usually on that component. In instances where the name server role of this entity requires mentioning, this memo uses the term "recursive name server". As an example of the difference, the name server component of a recursive name server receives DNS queries and the iterative resolver component sends queries.

The advent of IPv6 requires mentioning AAAA records as well as A records when discussing glue. To avoid continuous repetition and qualification, this memo uses the general term "address record" to encompass both A and AAAA records when a particular situation is relevant to both types.

1.2. Key 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 RFC 2119 [1].

2. Observed Iterative Resolver Misbehavior

2.1. Aggressive Requerying for Delegation Information

There can be times when every name server in a zone’s NS RRSet is unreachable (e.g., during a network outage), unavailable (e.g., the name server process is not running on the server host), or misconfigured (e.g., the name server is not authoritative for the given zone, also known as "lame"). Consider an iterative resolver that attempts to resolve a query for a domain name in such a zone and

discovers that none of the zone's name servers can provide an answer. We have observed a recursive name server implementation whose iterative resolver then verifies the zone's NS RRSet in its cache by querying for the zone's delegation information: it sends a query for the zone's NS RRSet to one of the parent zone's name servers. (Note that queries with QTYPE=NS are not required by the standard resolution algorithm described in Section 4.3.2 of RFC 1034 [2]. These NS queries represent this implementation's addition to that algorithm.)

For example, suppose that "example.com" has the following NS RRSet:

```
example.com.   IN   NS   ns1.example.com.
example.com.   IN   NS   ns2.example.com.
```

Upon receipt of a query for "www.example.com" and assuming that neither "ns1.example.com" nor "ns2.example.com" can provide an answer, this iterative resolver implementation immediately queries a "com" zone name server for the "example.com" NS RRSet to verify that it has the proper delegation information. This implementation performs this query to a zone's parent zone for each recursive query it receives that fails because of a completely unresponsive set of name servers for the target zone. Consider the effect when a popular zone experiences a catastrophic failure of all its name servers: now every recursive query for domain names in that zone sent to this recursive name server implementation results in a query to the failed zone's parent name servers. On one occasion when several dozen popular zones became unreachable, the query load on the com/net name servers increased by 50%.

We believe this verification query is not reasonable. Consider the circumstances: when an iterative resolver is resolving a query for a domain name in a zone it has not previously searched, it uses the list of name servers in the referral from the target zone's parent. If on its first attempt to search the target zone, none of the name servers in the referral is reachable, a verification query to the parent would be pointless: this query to the parent would come so quickly on the heels of the referral that it would be almost certain to contain the same list of name servers. The chance of discovering any new information is slim.

The other possibility is that the iterative resolver successfully contacts one of the target zone's name servers and then caches the NS RRSet from the authority section of a response, the proper behavior according to Section 5.4.1 of RFC 2181 [3], because the NS RRSet from the target zone is more trustworthy than delegation information from the parent zone. If, while processing a subsequent recursive query, the iterative resolver discovers that none of the name servers

specified in the cached NS RRSet is available or authoritative, querying the parent would be wrong. An NS RRSet from the parent zone would now be less trustworthy than data already in the cache.

For this query of the parent zone to be useful, the target zone's entire set of name servers would have to change AND the former set of name servers would have to be deconfigured or decommissioned AND the delegation information in the parent zone would have to be updated with the new set of name servers, all within the Time to Live (TTL) of the target zone's NS RRSet. We believe this scenario is uncommon: administrative best practices dictate that changes to a zone's set of name servers happen gradually when at all possible, with servers removed from the NS RRSet left authoritative for the zone as long as possible. The scenarios that we can envision that would benefit from the parent requery behavior do not outweigh its damaging effects.

This section should not be understood to claim that all queries to a zone's parent are bad. In some cases, such queries are not only reasonable but required. Consider the situation when required information, such as the address of a name server (i.e., the address record corresponding to the RDATA of an NS record), has timed out of an iterative resolver's cache before the corresponding NS record. If the name of the name server is below the apex of the zone, then the name server's address record is only available as glue in the parent zone. For example, consider this NS record:

```
example.com.        IN   NS   ns.example.com.
```

If a cache has this NS record but not the address record for "ns.example.com", it is unable to contact the "example.com" zone directly and must query the "com" zone to obtain the address record. Note, however, that such a query would not have QTYPE=NS according to the standard resolution algorithm.

2.1.1. Recommendation

An iterative resolver MUST NOT send a query for the NS RRSet of a non-responsive zone to any of the name servers for that zone's parent zone. For the purposes of this injunction, a non-responsive zone is defined as a zone for which every name server listed in the zone's NS RRSet:

1. is not authoritative for the zone (i.e., lame), or

2. returns a server failure response (RCODE=2), or

3. is dead or unreachable according to Section 7.2 of RFC 2308 [4].

2.2. Repeated Queries to Lame Servers

Section 2.1 describes a catastrophic failure: when every name server for a zone is unable to provide an answer for one reason or another. A more common occurrence is when a subset of a zone's name servers is unavailable or misconfigured. Different failure modes have different expected durations. Some symptoms indicate problems that are potentially transient, for example, various types of ICMP unreachable messages because a name server process is not running or a host or network is unreachable, or a complete lack of a response to a query. Such responses could be the result of a host rebooting or temporary outages; these events do not necessarily require any human intervention and can be reasonably expected to be temporary.

Other symptoms clearly indicate a condition requiring human intervention, such as lame server: if a name server is misconfigured and not authoritative for a zone delegated to it, it is reasonable to assume that this condition has potential to last longer than unreachability or unresponsiveness. Consequently, repeated queries to known lame servers are not useful. In this case of a condition with potential to persist for a long time, a better practice would be to maintain a list of known lame servers and avoid querying them repeatedly in a short interval.

It should also be noted, however, that some authoritative name server implementations appear to be lame only for queries of certain types as described in RFC 4074 [5]. In this case, it makes sense to retry the "lame" servers for other types of queries, particularly when all known authoritative name servers appear to be "lame".

2.2.1. Recommendation

Iterative resolvers SHOULD cache name servers that they discover are not authoritative for zones delegated to them (i.e., lame servers). If this caching is performed, lame servers MUST be cached against the specific query tuple <zone name, class, server IP address>. Zone name can be derived from the owner name of the NS record that was referenced to query the name server that was discovered to be lame.

Implementations that perform lame server caching MUST refrain from sending queries to known lame servers for a configurable time interval after the server is discovered to be lame. A minimum interval of thirty minutes is RECOMMENDED.

An exception to this recommendation occurs if all name servers for a zone are marked lame. In that case, the iterative resolver SHOULD temporarily ignore the servers' lameness status and query one or more servers. This behavior is a workaround for the type-specific lameness issue described in the previous section.

Implementors should take care not to make lame server avoidance logic overly broad: note that a name server could be lame for a parent zone but not a child zone, e.g., lame for "example.com" but properly authoritative for "sub.example.com". Therefore, a name server should not be automatically considered lame for subzones. In the case above, even if a name server is known to be lame for "example.com", it should be queried for QNAMEs at or below "sub.example.com" if an NS record indicates that it should be authoritative for that zone.

2.3. Inability to Follow Multiple Levels of Indirection

Some iterative resolver implementations are unable to follow sufficient levels of indirection. For example, consider the following delegations:

```
foo.example.        IN   NS   ns1.example.com.
foo.example.        IN   NS   ns2.example.com.

example.com.        IN   NS   ns1.test.example.net.
example.com.        IN   NS   ns2.test.example.net.

test.example.net.   IN   NS   ns1.test.example.net.
test.example.net.   IN   NS   ns2.test.example.net.
```

An iterative resolver resolving the name "www.foo.example" must follow two levels of indirection, first obtaining address records for "ns1.test.example.net" or "ns2.test.example.net" in order to obtain address records for "ns1.example.com" or "ns2.example.com" in order to query those name servers for the address records of "www.foo.example". Although this situation may appear contrived, we have seen multiple similar occurrences and expect more as new generic top-level domains (gTLDs) become active. We anticipate many zones in new gTLDs will use name servers in existing gTLDs, increasing the number of delegations using out-of-zone name servers.

2.3.1. Recommendation

Clearly constructing a delegation that relies on multiple levels of indirection is not a good administrative practice. However, the practice is widespread enough to require that iterative resolvers be able to cope with it. Iterative resolvers SHOULD be able to handle arbitrary levels of indirection resulting from out-of-zone name

servers. Iterative resolvers SHOULD implement a level-of-effort counter to avoid loops or otherwise performing too much work in resolving pathological cases.

A best practice that avoids this entire issue of indirection is to name one or more of a zone's name servers in the zone itself. For example, if the zone is named "example.com", consider naming some of the name servers "ns{1,2,...}.example.com" (or similar).

2.4. Aggressive Retransmission when Fetching Glue

When an authoritative name server responds with a referral, it includes NS records in the authority section of the response. According to the algorithm in Section 4.3.2 of RFC 1034 [2], the name server should also "put whatever addresses are available into the additional section, using glue RRs if the addresses are not available from authoritative data or the cache." Some name server implementations take this address inclusion a step further with a feature called "glue fetching". A name server that implements glue fetching attempts to include address records for every NS record in the authority section. If necessary, the name server issues multiple queries of its own to obtain any missing address records.

Problems with glue fetching can arise in the context of "authoritative-only" name servers, which only serve authoritative data and ignore requests for recursion. Such an entity will not normally generate any queries of its own. Instead it answers non-recursive queries from iterative resolvers looking for information in zones it serves. With glue fetching enabled, however, an authoritative server invokes an iterative resolver to look up an unknown address record to complete the additional section of a response.

We have observed situations where the iterative resolver of a glue-fetching name server can send queries that reach other name servers, but is apparently prevented from receiving the responses. For example, perhaps the name server is authoritative-only and therefore its administrators expect it to receive only queries and not responses. Perhaps unaware of glue fetching and presuming that the name server's iterative resolver will generate no queries, its administrators place the name server behind a network device that prevents it from receiving responses. If this is the case, all glue-fetching queries will go unanswered.

We have observed name server implementations whose iterative resolvers retry excessively when glue-fetching queries are unanswered. A single com/net name server has received hundreds of queries per second from a single such source. Judging from the

specific queries received and based on additional analysis, we believe these queries result from overly aggressive glue fetching.

2.4.1. Recommendation

Implementers whose name servers support glue fetching SHOULD take care to avoid sending queries at excessive rates. Implementations SHOULD support throttling logic to detect when queries are sent but no responses are received.

2.5. Aggressive Retransmission behind Firewalls

A common occurrence and one of the largest sources of repeated queries at the com/net and root name servers appears to result from resolvers behind misconfigured firewalls. In this situation, an iterative resolver is apparently allowed to send queries through a firewall to other name servers, but not receive the responses. The result is more queries than necessary because of retransmission, all of which are useless because the responses are never received. Just as with the glue-fetching scenario described in Section 2.4, the queries are sometimes sent at excessive rates. To make matters worse, sometimes the responses, sent in reply to legitimate queries, trigger an alarm on the originator's intrusion detection system. We are frequently contacted by administrators responding to such alarms who believe our name servers are attacking their systems.

Not only do some resolvers in this situation retransmit queries at an excessive rate, but they continue to do so for days or even weeks. This scenario could result from an organization with multiple recursive name servers, only a subset of whose iterative resolvers' traffic is improperly filtered in this manner. Stub resolvers in the organization could be configured to query multiple recursive name servers. Consider the case where a stub resolver queries a filtered recursive name server first. The iterative resolver of this recursive name server sends one or more queries whose replies are filtered, so it cannot respond to the stub resolver, which times out. Then the stub resolver retransmits to a recursive name server that is able to provide an answer. Since resolution ultimately succeeds the underlying problem might not be recognized or corrected. A popular stub resolver implementation has a very aggressive retransmission schedule, including simultaneous queries to multiple recursive name servers, which could explain how such a situation could persist without being detected.

2.5.1. Recommendation

The most obvious recommendation is that administrators SHOULD take care not to place iterative resolvers behind a firewall that allows queries, but not the resulting replies, to pass through.

Iterative resolvers SHOULD take care to avoid sending queries at excessive rates. Implementations SHOULD support throttling logic to detect when queries are sent but no responses are received.

2.6. Misconfigured NS Records

Sometimes a zone administrator forgets to add the trailing dot on the domain names in the RDATA of a zone's NS records. Consider this fragment of the zone file for "example.com":

```
$ORIGIN example.com.
example.com.      3600   IN   NS   ns1.example.com  ; Note missing
example.com.      3600   IN   NS   ns2.example.com  ; trailing dots
```

The zone's authoritative servers will parse the NS RDATA as "ns1.example.com.example.com" and "ns2.example.com.example.com" and return NS records with this incorrect RDATA in responses, including typically the authority section of every response containing records from the "example.com" zone.

Now consider a typical sequence of queries. An iterative resolver attempting to resolve address records for "www.example.com" with no cached information for this zone will query a "com" authoritative server. The "com" server responds with a referral to the "example.com" zone, consisting of NS records with valid RDATA and associated glue records. (This example assumes that the "example.com" zone delegation information is correct in the "com" zone.) The iterative resolver caches the NS RRSet from the "com" server and follows the referral by querying one of the "example.com" authoritative servers. This server responds with the "www.example.com" address record in the answer section and, typically, the "example.com" NS records in the authority section and, if space in the message remains, glue address records in the additional section. According to Section 5.4.1 of RFC 2181 [3], NS records in the authority section of an authoritative answer are more trustworthy than NS records from the authority section of a non-authoritative answer. Thus, the "example.com" NS RRSet just received from the "example.com" authoritative server overrides the "example.com" NS RRSet received moments ago from the "com" authoritative server.

But the "example.com" zone contains the erroneous NS RRSet as shown in the example above. Subsequent queries for names in "example.com" will cause the iterative resolver to attempt to use the incorrect NS records and so it will try to resolve the nonexistent names "ns1.example.com.example.com" and "ns2.example.com.example.com". In this example, since all of the zone's name servers are named in the zone itself (i.e., "ns1.example.com.example.com" and "ns2.example.com.example.com" both end in "example.com") and all are bogus, the iterative resolver cannot reach any "example.com" name servers. Therefore, attempts to resolve these names result in address record queries to the "com" authoritative servers. Queries for such obviously bogus glue address records occur frequently at the com/net name servers.

2.6.1. Recommendation

An authoritative server can detect this situation. A trailing dot missing from an NS record's RDATA always results by definition in a name server name that exists somewhere under the apex of the zone that the NS record appears in. Note that further levels of delegation are possible, so a missing trailing dot could inadvertently create a name server name that actually exists in a subzone.

An authoritative name server SHOULD issue a warning when one of a zone's NS records references a name server below the zone's apex when a corresponding address record does not exist in the zone AND there are no delegated subzones where the address record could exist.

2.7. Name Server Records with Zero TTL

Sometimes a popular com/net subdomain's zone is configured with a TTL of zero on the zone's NS records, which prohibits these records from being cached and will result in a higher query volume to the zone's authoritative servers. The zone's administrator should understand the consequences of such a configuration and provision resources accordingly. A zero TTL on the zone's NS RRSet, however, carries additional consequences beyond the zone itself: if an iterative resolver cannot cache a zone's NS records because of a zero TTL, it will be forced to query that zone's parent's name servers each time it resolves a name in the zone. The com/net authoritative servers do see an increased query load when a popular com/net subdomain's zone is configured with a TTL of zero on the zone's NS records.

A zero TTL on an RRSet expected to change frequently is extreme but permissible. A zone's NS RRSet is a special case, however, because changes to it must be coordinated with the zone's parent. In most zone parent/child relationships that we are aware of, there is

typically some delay involved in effecting changes. Furthermore, changes to the set of a zone's authoritative name servers (and therefore to the zone's NS RRSet) are typically relatively rare: providing reliable authoritative service requires a reasonably stable set of servers. Therefore, an extremely low or zero TTL on a zone's NS RRSet rarely makes sense, except in anticipation of an upcoming change. In this case, when the zone's administrator has planned a change and does not want iterative resolvers throughout the Internet to cache the NS RRSet for a long period of time, a low TTL is reasonable.

2.7.1. Recommendation

Because of the additional load placed on a zone's parent's authoritative servers resulting from a zero TTL on a zone's NS RRSet, under such circumstances authoritative name servers SHOULD issue a warning when loading a zone.

2.8. Unnecessary Dynamic Update Messages

The UPDATE message specified in RFC 2136 [6] allows an authorized agent to update a zone's data on an authoritative name server using a DNS message sent over the network. Consider the case of an agent desiring to add a particular resource record. Because of zone cuts, the agent does not necessarily know the proper zone to which the record should be added. The dynamic update process requires that the agent determine the appropriate zone so the UPDATE message can be sent to one of the zone's authoritative servers (typically the primary master as specified in the zone's Start of Authority (SOA) record's MNAME field).

The appropriate zone to update is the closest enclosing zone, which cannot be determined only by inspecting the domain name of the record to be updated, since zone cuts can occur anywhere. One way to determine the closest enclosing zone entails walking up the name space tree by sending repeated UPDATE messages until successful. For example, consider an agent attempting to add an address record with the name "foo.bar.example.com". The agent could first attempt to update the "foo.bar.example.com" zone. If the attempt failed, the update could be directed to the "bar.example.com" zone, then the "example.com" zone, then the "com" zone, and finally the root zone.

A popular dynamic agent follows this algorithm. The result is many UPDATE messages received by the root name servers, the com/net authoritative servers, and presumably other TLD authoritative servers. A valid question is why the algorithm proceeds to send updates all the way to TLD and root name servers. This behavior is not entirely unreasonable: in enterprise DNS architectures with an

"internal root" design, there could conceivably be private, non-public TLD or root zones that would be the appropriate targets for a dynamic update.

A significant deficiency with this algorithm is that knowledge of a given UPDATE message's failure is not helpful in directing future UPDATE messages to the appropriate servers. A better algorithm would be to find the closest enclosing zone by walking up the name space with queries for SOA or NS rather than "probing" with UPDATE messages. Once the appropriate zone is found, an UPDATE message can be sent. In addition, the results of these queries can be cached to aid in determining the closest enclosing zones for future updates. Once the closest enclosing zone is determined with this method, the update will either succeed or fail and there is no need to send further updates to higher-level zones. The important point is that walking up the tree with queries yields cacheable information, whereas walking up the tree by sending UPDATE messages does not.

2.8.1. Recommendation

Dynamic update agents SHOULD send SOA or NS queries to progressively higher-level names to find the closest enclosing zone for a given name to update. Only after the appropriate zone is found should the client send an UPDATE message to one of the zone's authoritative servers. Update clients SHOULD NOT "probe" using UPDATE messages by walking up the tree to progressively higher-level zones.

2.9. Queries for Domain Names Resembling IPv4 Addresses

The root name servers receive a significant number of A record queries where the QNAME looks like an IPv4 address. The source of these queries is unknown. It could be attributed to situations where a user believes that an application will accept either a domain name or an IP address in a given configuration option. The user enters an IP address, but the application assumes that any input is a domain name and attempts to resolve it, resulting in an A record lookup. There could also be applications that produce such queries in a misguided attempt to reverse map IP addresses.

These queries result in Name Error (RCODE=3) responses. An iterative resolver can negatively cache such responses, but each response requires a separate cache entry; i.e., a negative cache entry for the domain name "192.0.2.1" does not prevent a subsequent query for the domain name "192.0.2.2".

2.9.1. Recommendation

It would be desirable for the root name servers not to have to answer these queries: they unnecessarily consume CPU resources and network bandwidth. A possible solution is to delegate these numeric TLDs from the root zone to a separate set of servers to absorb the traffic. The "black hole servers" used by the AS 112 Project (http://www.as112.net), which are currently delegated the in-addr.arpa zones corresponding to RFC 1918 [7] private use address space, would be a possible choice to receive these delegations. Of course, the proper and usual root zone change procedures would have to be followed to make such a change to the root zone.

2.10. Misdirected Recursive Queries

The root name servers receive a significant number of recursive queries (i.e., queries with the Recursion Desired (RD) bit set in the header). Since none of the root servers offers recursion, the servers' response in such a situation ignores the request for recursion and the response probably does not contain the data the querier anticipated. Some of these queries result from users configuring stub resolvers to query a root server. (This situation is not hypothetical: we have received complaints from users when this configuration does not work as hoped.) Of course, users should not direct stub resolvers to use name servers that do not offer recursion, but we are not aware of any stub resolver implementation that offers any feedback to the user when so configured, aside from simply "not working".

2.10.1. Recommendation

When the IP address of a name server that supposedly offers recursion is configured in a stub resolver using an interactive user interface, the resolver could send a test query to verify that the server indeed supports recursion (i.e., verify that the response has the RA bit set in the header). The user could be notified immediately if the server is non-recursive.

The stub resolver could also report an error, either through a user interface or in a log file, if the queried server does not support recursion. Error reporting SHOULD be throttled to avoid a notification or log message for every response from a non-recursive server.

2.11. Suboptimal Name Server Selection Algorithm

An entire document could be devoted to the topic of problems with different implementations of the recursive resolution algorithm. The entire process of recursion is woefully under-specified, requiring each implementor to design an algorithm. Sometimes implementors make poor design choices that could be avoided if a suggested algorithm and best practices were documented, but that is a topic for another document.

Some deficiencies cause significant operational impact and are therefore worth mentioning here. One of these is name server selection by an iterative resolver. When an iterative resolver wants to contact one of a zone's authoritative name servers, how does it choose from the NS records listed in the zone's NS RRSet? If the selection mechanism is suboptimal, queries are not spread evenly among a zone's authoritative servers. The details of the selection mechanism are up to the implementor, but we offer some suggestions.

2.11.1. Recommendation

This list is not conclusive, but reflects the changes that would produce the most impact in terms of reducing disproportionate query load among a zone's authoritative servers. That is, these changes would help spread the query load evenly.

- Do not make assumptions based on NS RRSet order: all NS RRs SHOULD be treated equally. (In the case of the "com" zone, for example, most of the root servers return the NS record for "a.gtld-servers.net" first in the authority section of referrals. Apparently as a result, this server receives disproportionately more traffic than the other twelve authoritative servers for "com".)

- Use all NS records in an RRSet. (For example, we are aware of implementations that hard-coded information for a subset of the root servers.)

- Maintain state and favor the best-performing of a zone's authoritative servers. A good definition of performance is response time. Non-responsive servers can be penalized with an extremely high response time.

- Do not lock onto the best-performing of a zone's name servers. An iterative resolver SHOULD periodically check the performance of all of a zone's name servers to adjust its determination of the best-performing one.

3. Security Considerations

The iterative resolver misbehavior discussed in this document exposes the root and TLD name servers to increased risk of both intentional and unintentional Denial of Service attacks.

We believe that implementation of the recommendations offered in this document will reduce the amount of unnecessary traffic seen at root and TLD name servers, thus reducing the opportunity for an attacker to use such queries to his or her advantage.

4. Acknowledgements

The authors would like to thank the following people for their comments that improved this document: Andras Salamon, Dave Meyer, Doug Barton, Jaap Akkerhuis, Jinmei Tatuya, John Brady, Kevin Darcy, Olafur Gudmundsson, Pekka Savola, Peter Koch, and Rob Austein. We apologize if we have omitted anyone; any oversight was unintentional.

5. Internationalization Considerations

There are no new internationalization considerations introduced by this memo.

6. References

6.1. Normative References

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

[2] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

6.2. Informative References

[3] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.

[4] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, March 1998.

[5] Morishita, Y. and T. Jinmei, "Common Misbehavior Against DNS Queries for IPv6 Addresses", RFC 4074, May 2005.

[6] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[7] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

Authors' Addresses

Matt Larson
VeriSign, Inc.
21345 Ridgetop Circle
Dulles, VA 20166-6503
USA

EMail: mlarson@verisign.com

Piet Barber
VeriSign, Inc.
21345 Ridgetop Circle
Dulles, VA 20166-6503
USA

EMail: pbarber@verisign.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).