Internet Engineering Task Force (IETF)
Request for Comments: 8198
Updates: 4035
Category: Standards Track
ISSN: 2070-1721

K. Fujiwara
JPRS
A. Kato
Keio/WIDE
W. Kumari
Google

July 2017

Aggressive Use of DNSSEC-Validated Cache

Abstract

The DNS relies upon caching to scale; however, the cache lookup generally requires an exact match. This document specifies the use of NSEC/NSEC3 resource records to allow DNSSEC-validating resolvers to generate negative answers within a range and positive answers from wildcards. This increases performance, decreases latency, decreases resource utilization on both authoritative and recursive servers, and increases privacy. Also, it may help increase resilience to certain DoS attacks in some circumstances.

This document updates RFC 4035 by allowing validating resolvers to generate negative answers based upon NSEC/NSEC3 records and positive answers in the presence of wildcards.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

A DNS negative cache exists, and is used to cache the fact that an RRset does not exist. This method of negative caching requires exact matching; this leads to unnecessary additional lookups, increases latency, leads to extra resource utilization on both authoritative and recursive servers, and decreases privacy by leaking queries.

This document updates RFC 4035 to allow resolvers to use NSEC/NSEC3 resource records to synthesize negative answers from the information they have in the cache. This allows validating resolvers to respond with a negative answer immediately if the name in question falls into a range expressed by an NSEC/NSEC3 resource record already in the cache. It also allows the synthesis of positive answers in the presence of wildcard records.

Aggressive negative caching was first proposed in Section 6 of DNSSEC Lookaside Validation (DLV) [RFC5074] in order to find covering NSEC records efficiently.

[RFC8020] and [RES-IMPROVE] propose steps to using NXDOMAIN information for more effective caching. This document takes this technique further.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Many of the specialized terms used in this document are defined in DNS Terminology [RFC7719].

The key words "source of synthesis" in this document are to be interpreted as described in [RFC4592].

3. Problem Statement

The DNS negative cache caches negative (non-existent) information, and requires an exact match in most instances [RFC2308].

Assume that the (DNSSEC-signed) "example.com" zone contains:

```
albatross.example.com. IN A 192.0.2.1
elephant.example.com.  IN A 192.0.2.2
zebra.example.com.     IN A 192.0.2.3
```

If a validating resolver receives a query for cat.example.com, it contacts its resolver (which may be itself) to query the example.com servers and will get back an NSEC record stating that there are no records (alphabetically) between albatross and elephant, or an NSEC3 record stating there is nothing between two hashed names. The resolver then knows that cat.example.com does not exist; however, it does not use the fact that the proof covers a range (albatross to elephant) to suppress queries for other labels that fall within this range. This means that if the validating resolver gets a query for ball.example.com (or dog.example.com) it will once again go off and query the example.com servers for these names.

Apart from wasting bandwidth, this also wastes resources on the recursive server (it needs to keep state for outstanding queries), wastes resources on the authoritative server (it has to answer additional questions), increases latency (the end user has to wait longer than necessary to get back an NXDOMAIN answer), can be used by attackers to cause a DoS, and also has privacy implications (e.g., typos leak out further than necessary).

Another example: assume that the (DNSSEC-signed) "example.org" zone contains:

```
avocado.example.org.   IN A 192.0.2.1
*.example.org.         IN A 192.0.2.2
zucchini.example.org.  IN A 192.0.2.3
```

If a query is received for leek.example.org, the system contacts its resolver (which may be itself) to query the example.org servers and will get back an NSEC record stating that there are no records (alphabetically) between avocado and zucchini (or an NSEC3 record stating there is nothing between two hashed names), as well as an answer for leek.example.org, with the label count of the signature set to two (see [RFC7129], Section 5.3 for more details).

If the validating resolver gets a query for banana.example.org, it will once again go off and query the example.org servers for banana.example.org (even though it already has proof that there is a wildcard record) -- just like above, this has privacy implications, wastes resources, can be used to contribute to a DoS, etc.

4. Background

DNSSEC [RFC4035] and [RFC5155] both provide "authenticated denial of existence"; this is a cryptographic proof that the queried-for name does not exist or the type does not exist. Proof that a name does not exist is accomplished by providing a (DNSSEC-secured) record containing the names that appear alphabetically before and after the queried-for name. In the first example above, if the (DNSSEC-validating) recursive server were to query for dog.example.com, it would receive a (signed) NSEC record stating that there are no labels between "albatross" and "elephant" (or, for NSEC3, a similar pair of hashed names). This is a signed, cryptographic proof that these names are the ones before and after the queried-for label. As dog.example.com falls within this range, the recursive server knows that dog.example.com really does not exist. Proof that a type does not exist is accomplished by providing a (DNSSEC-secured) record containing the queried-for name, and a type bitmap that does not include the requested type.

This document specifies that this NSEC/NSEC3 record should be used to generate negative answers for any queries that the validating server receives that fall within the range covered by the record (for the TTL for the record). This document also specifies that a positive answer should be generated for any queries that the validating server receives that are proven to be covered by a wildcard record.

Section 4.5 of [RFC4035] says:

> In theory, a resolver could use wildcards or NSEC RRs to generate positive and negative responses (respectively) until the TTL or signatures on the records in question expire. However, it seems prudent for resolvers to avoid blocking new authoritative data or synthesizing new data on their own. Resolvers that follow this recommendation will have a more consistent view of the namespace.

And, earlier, Section 4.5 of [RFC4035] says:

> The reason for these recommendations is that, between the initial query and the expiration of the data from the cache, the authoritative data might have been changed (for example, via dynamic update).

In other words, if a resolver generates negative answers from an NSEC record, it will not send any queries for names within that NSEC range (for the TTL). If a new name is added to the zone during this interval, the resolver will not know this. Similarly, if the resolver is generating responses from a wildcard record, it will continue to do so (for the TTL).

We believe that this recommendation can be relaxed because, in the absence of this technique, a lookup for the exact name could have come in during this interval, and so a negative answer could already be cached (see [RFC2308] for more background). This means that zone operators should have no expectation that an added name would work immediately. With DNSSEC and aggressive use of DNSSEC-validated cache, the TTL of the NSEC/NSEC3 record and the SOA.MINIMUM field are the authoritative statement of how quickly a name can start working within a zone.

5. Aggressive Use of DNSSEC-Validated Cache

This document relaxes the restriction given in Section 4.5 of [RFC4035]. See Section 7 for more detail.

If the negative cache of the validating resolver has sufficient information to validate the query, the resolver SHOULD use NSEC, NSEC3, and wildcard records to synthesize answers as described in this document. Otherwise, it MUST fall back to send the query to the authoritative DNS servers.

5.1. NSEC

The validating resolver needs to check the existence of an NSEC RR matching/covering the source of synthesis and an NSEC RR covering the query name.

If denial of existence can be determined according to the rules set out in Section 5.4 of [RFC4035], using NSEC records in the cache, then the resolver can immediately return an NXDOMAIN or NODATA (as appropriate) response.

5.2. NSEC3

NSEC3 aggressive negative caching is more difficult than NSEC aggressive caching. If the zone is signed with NSEC3, the validating resolver needs to check the existence of non-terminals and wildcards that derive from query names.

If denial of existence can be determined according to the rules set out in [RFC5155], Sections 8.4, 8.5, 8.6, and 8.7, using NSEC3 records in the cache, then the resolver can immediately return an NXDOMAIN or NODATA response (as appropriate).

If a covering NSEC3 RR has an Opt-Out flag, the covering NSEC3 RR does not prove the non-existence of the domain name and the aggressive negative caching is not possible for the domain name.

5.3. Wildcards

The last paragraph of [RFC4035], Section 4.5 also discusses the use of wildcards and NSEC RRs to generate positive responses and recommends that it not be relied upon. Just like the case for the aggressive use of NSEC/NSEC3 for negative answers, we revise this recommendation.

As long as the validating resolver can determine that a name would not exist without the wildcard match, determined according to the rules set out in Section 5.3.4 of [RFC4035] (NSEC), or in Section 8.8 of [RFC5155], it SHOULD synthesize an answer (or NODATA response) for that name using the cache-deduced wildcard. If the corresponding wildcard record is not in the cache, it MUST fall back to send the query to the authoritative DNS servers.

5.4. Consideration on TTL

The TTL value of negative information is especially important, because newly added domain names cannot be used while the negative information is effective.

Section 5 of [RFC2308] suggests a maximum default negative cache TTL value of 3 hours (10800). It is RECOMMENDED that validating resolvers limit the maximum effective TTL value of negative responses (NSEC/NSEC3 RRs) to this same value.

Section 5 of [RFC2308] also states that a negative cache entry TTL is taken from the minimum of the SOA.MINIMUM field and SOA's TTL. This can be less than the TTL of an NSEC or NSEC3 record, since their TTL is equal to the SOA.MINIMUM field (see [RFC4035], Section 2.3 and [RFC5155], Section 3).

A resolver that supports aggressive use of NSEC and NSEC3 SHOULD reduce the TTL of NSEC and NSEC3 records to match the SOA.MINIMUM field in the authority section of a negative response, if SOA.MINIMUM is smaller.

6. Benefits

The techniques described in this document provide a number of benefits, including (in no specific order):

Reduced latency: By answering directly from cache, validating resolvers can immediately inform clients that the name they are looking for does not exist, improving the user experience.

Decreased recursive server load: By answering queries from the cache by synthesizing answers, validating servers avoid having to send a query and wait for a response. In addition to decreasing the bandwidth used, it also means that the server does not need to allocate and maintain state, thereby decreasing memory and CPU load.

Decreased authoritative server load: Because recursive servers can answer queries without asking the authoritative server, the authoritative servers receive fewer queries. This decreases the authoritative server bandwidth, queries per second, and CPU utilization.

The scale of the benefit depends upon multiple factors, including the query distribution. For example, at the time of this writing, around 65% of queries to root name servers result in NXDOMAIN responses (see statistics from [ROOT-SERVERS]); this technique will eliminate a sizable quantity of these.

The technique described in this document may also mitigate so-called "random QNAME attacks", in which attackers send many queries for random subdomains to resolvers. As the resolver will not have the answers cached, it has to ask external servers for each random query, leading to a DoS on the authoritative servers (and often resolvers). The technique may help mitigate these attacks by allowing the resolver to answer directly from the cache for any random queries that fall within already requested ranges. It will not always work as an effective defense, not least because not many zones are DNSSEC signed at all -- but it will still provide an additional layer of defense.

As these benefits are only accrued by those using DNSSEC, it is hoped that these techniques will lead to more DNSSEC deployment.

7. Update to RFC 4035

Section 4.5 of [RFC4035] shows that "In theory, a resolver could use wildcards or NSEC RRs to generate positive and negative responses (respectively) until the TTL or signatures on the records in question expire. However, it seems prudent for resolvers to avoid blocking new authoritative data or synthesizing new data on their own. Resolvers that follow this recommendation will have a more consistent view of the namespace".

The paragraph is updated as follows:

```
+--------------------------------------------------------------------+
|  Once the records are validated, DNSSEC-enabled validating          |
|  resolvers SHOULD use wildcards and NSEC/NSEC3 resource records     |
|  to generate positive and negative responses until the              |
|  effective TTLs or signatures for those records expire.             |
+--------------------------------------------------------------------+
```

8. IANA Considerations

This document does not require any IANA actions.

9. Security Considerations

Use of NSEC/NSEC3 resource records without DNSSEC validation may create serious security issues, and so this technique requires DNSSEC validation.

Newly registered resource records may not be used immediately. However, choosing a suitable TTL value and a negative cache TTL value (SOA.MINIMUM field) will mitigate the delay concern, and it is not a security problem.

It is also suggested to limit the maximum TTL value of NSEC/NSEC3 resource records in the negative cache to, for example, 10800 seconds (3 hours), to mitigate this issue.

Although the TTL of NSEC/NSEC3 records is typically fairly short (minutes or hours), their RRSIG expiration time can be much further in the future (weeks). An attacker who is able to successfully spoof responses might poison a cache with old NSEC/NSEC3 records. If the resolver is not making aggressive use of NSEC/NSEC3, the attacker has to repeat the attack for every query. If the resolver is making aggressive use of NSEC/NSEC3, one successful attack would be able to suppress many queries for new names, up to the negative TTL.

10. References

10.1. Normative References

[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>.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <http://www.rfc-editor.org/info/rfc2308>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <http://www.rfc-editor.org/info/rfc4035>.

[RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name System", RFC 4592, DOI 10.17487/RFC4592, July 2006, <http://www.rfc-editor.org/info/rfc4592>.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <http://www.rfc-editor.org/info/rfc5155>.

[RFC7129] Gieben, R. and W. Mekking, "Authenticated Denial of Existence in the DNS", RFC 7129, DOI 10.17487/RFC7129, February 2014, <http://www.rfc-editor.org/info/rfc7129>.

[RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <http://www.rfc-editor.org/info/rfc7719>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <http://www.rfc-editor.org/info/rfc8174>.

10.2. Informative References

[RES-IMPROVE] Vixie, P., Joffe, R., and F. Neves, "Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness", Work in Progress, draft-vixie-dnsext-resimprove-00, June 2010.

[RFC5074] Weiler, S., "DNSSEC Lookaside Validation (DLV)", RFC 5074, DOI 10.17487/RFC5074, November 2007, <http://www.rfc-editor.org/info/rfc5074>.

[RFC8020] Bortzmeyer, S. and S. Huque, "NXDOMAIN: There Really Is Nothing Underneath", RFC 8020, DOI 10.17487/RFC8020, November 2016, <http://www.rfc-editor.org/info/rfc8020>.

[ROOT-SERVERS] "Root Server Technical Operations Assn", <http://www.root-servers.org/>.

Appendix A. Detailed Implementation Notes

- Previously, cached negative responses were indexed by QNAME, QCLASS, QTYPE, and the setting of the CD bit (see RFC 4035, Section 4.7), and only queries matching the index key would be answered from the cache. With aggressive negative caching, the validator, in addition to checking to see if the answer is in its cache before sending a query, checks to see whether any cached and validated NSEC record denies the existence of the sought record(s). Using aggressive negative caching, a validator will not make queries for any name covered by a cached and validated NSEC record. Furthermore, a validator answering queries from clients will synthesize a negative answer (or NODATA response) whenever it has an applicable validated NSEC in its cache unless the CD bit was set on the incoming query. (Imported from Section 6 of [RFC5074].)

- Implementing aggressive negative caching suggests that a validator will need to build an ordered data structure of NSEC and NSEC3 records for each signer domain name of NSEC/NSEC3 records in order to efficiently find covering NSEC/NSEC3 records. Call the table as "NSEC_TABLE". (Imported from Section 6.1 of [RFC5074] and expanded.)

- The aggressive negative caching may be inserted at the cache lookup part of the recursive resolvers.

- If errors happen in an aggressive negative caching algorithm, resolvers MUST fall back to resolve the query as usual. "Resolve the query as usual" means that the resolver must process the query as though it does not implement aggressive negative caching.

Appendix B. Procedure for Determining ENT vs. NXDOMAIN with NSEC

This procedure outlines how to determine if a given name does not exist, or is an ENT (empty non-terminal; see [RFC5155], Section 1.3) with NSEC.

If the NSEC record has not been verified as secure, discard it.

If the given name sorts before or matches the NSEC owner name, discard it as it does not prove the NXDOMAIN or ENT.

If the given name is a subdomain of the NSEC owner name and the NS bit is present and the SOA bit is absent, then discard the NSEC as it is from a parent zone.

If the next domain name sorts after the NSEC owner name and the given name sorts after or matches next domain name, then discard the NSEC record as it does not prove the NXDOMAIN or ENT.

If the next domain name sorts before or matches the NSEC owner name and the given name is not a subdomain of the next domain name, then discard the NSEC as it does not prove the NXDOMAIN or ENT.

You now have an NSEC record that proves the NXDOMAIN or ENT.

If the next domain name is a subdomain of the given name, you have an ENT. Otherwise, you have an NXDOMAIN.

Acknowledgments

The authors gratefully acknowledge DNSSEC Lookaside Validation (DLV) [RFC5074] author Samuel Weiler and the Unbound developers.

Thanks to Mark Andrews for providing the helpful notes for implementors provided in Appendix B.

The authors would like to specifically thank Stephane Bortzmeyer (for standing next to and helping edit), Ralph Dolmans, Tony Finch, Tatuya JINMEI for extensive review and comments, and also Mark Andrews, Casey Deccio, Alexander Dupuy, Olafur Gudmundsson, Bob Harold, Shumon Huque, John Levine, Pieter Lexis, Matthijs Mekking (who even sent pull requests!), and Ondrej Sury.

Authors' Addresses

Kazunori Fujiwara
Japan Registry Services Co., Ltd.
Chiyoda First Bldg. East 13F, 3-8-1 Nishi-Kanda
Chiyoda-ku, Tokyo 101-0065
Japan

Phone: +81 3 5215 8451
Email: fujiwara@jprs.co.jp

Akira Kato
Keio University/WIDE Project
Graduate School of Media Design, 4-1-1 Hiyoshi
Kohoku, Yokohama 223-8526
Japan

Phone: +81 45 564 2490
Email: kato@wide.ad.jp

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
United States of America

Email: warren@kumari.net