Network Working Group
Internet-Draft
Updates: 4035 (if approved)
Intended status: Informational
Expires: December 29, 2016

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

June 27, 2016

Aggressive use of NSEC/NSEC3
draft-ietf-dnsop-nsec-aggressiveuse-00

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 generate negative answers within a range. This increases resilience to DoS attacks, increases performance / decreases latency, decreases resource utilization on both authoritative and recursive servers, and also increases privacy.

This document updates RFC4035 by allowing resolvers to generate negative answers based upon NSEC/NSEC3 records.

[Ed note: Text inside square brackets ([]) is additional background information, answers to frequently asked questions, general musings, etc. They will be removed before publication.This document is being collaborated on in Github at: https://github.com/wkumari/draft-ietf-dnsop-nsec-aggressiveuse. The most recent version of the document, open issues, etc should all be available here. The authors (gratefully) accept pull requests.

Known / open issues [To be moved to Github issue tracker]:

1. We say things like: "Currently the DNS does ..." - this will not be true after this is deployed, but I'm having a hard time rewording this. "Without the techniques described in this document..." seems klunky. Perhaps "historically?!"

2. We currently say this SHOULD be enabled by default. Is that what the working group wants, or should this be an implementation choice?

]

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

A DNS negative cache currently exists, and is used to cache the fact that a name does not exist. This method of negative caching requires exact matching; this leads to unnecessary additional lookups, which have negative implications for DoS survivability, 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 recursive resolvers to use NSEC/NSEC3 resource records to aggressively cache negative answers. This would allow such resolvers to respond with NXDOMAIN immediately if the name in question falls into a range expressed by a NSEC/NSEC3 resource record already in the cache.

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

Section 3 of [I-D.vixie-dnsext-resimprove] "Stopping Downward Cache Search on NXDOMAIN" and [I-D.ietf-dnsop-nxdomain-cut] proposed another approach to use NXDOMAIN information effectively.

2. Terminology

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 [RFC2119].

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

The key words "Closest Encloser" and "Source of Synthesis" in this document are to be interpreted as described in[RFC4592].

"Closest Encloser" is also defined in NSEC3 [RFC5155], as is "Next closer name".

3. Problem Statement

The current 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:

```
apple.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 recursive resolver gets a query for cat.example.com, it will query the example.com authoritative servers and will get back an NSEC (or NSEC3) record starting that there are no records between apple and elephant. The recursive resolver then knows that cat.example.com does not exist; however, it (currently) does not use the fact that the proof covers a range (apple to elephant) to suppress queries for other labels that fall within this range. This means that if the recursive resolvers 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 (see additional resources), and also has privacy implications (e.g: typos leak out further than necessary).

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, accomplished by providing a (DNSSEC secured) record containing the names which appear alphabetically before and after the queried for name. In the example above, if the (DNSSEC validating) recursive server were to query for lion.example.com it would receive a (signed) NSEC/NSEC3 record stating that there are no labels between "elephant" and "zebra". This is a signed, cryptographic proof that

these names are the ones before and after the queried for label. As lion.example.com falls within this range, the recursive server knows that lion.example.com really does not exist. This document specifies that this NSEC/NSEC3 record should be used to generate negative answers for any queries that the recursive server receives that fall within the range covered by the record (for the TTL for the record).

[RFC4035]; Section 4.5 states:

For a zone signed with NSEC, it would be possible to use the information carried in NSEC resource records to indicate the non-existence of a range of names. However, such use is discouraged by Section 4.5 of RFC4035. It is recommended that readers read RFC4035 in its entirety for a better understanding. At the root of the concern is that new records could have been added to the zone during the TTL of the NSEC record, and that generating negative responses from the NSEC record would hide these. We believe this recommendation can be relaxed because lookups for the specific name could have come in during the normal negative cache time and so operators should have no expectation that an added name would work immediately. We think that the TTL of the NSEC record is the authoritive statement of how quickly a name can start working within a zone.

5. Proposed Solution

5.1. Aggressive Negative Caching

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".

To reduce non-existent queries sent to authoritative DNS servers, this restriction could be relaxed, as follows:

```
+--------------------------------------------------------------+
|  Once the records are validated, DNSSEC enabled full-service |
|  resolvers MAY use NSEC/NSEC3 resource records to generate   |
|  negative responses until their effective TTLs or signatures |
|  for those records expire.                                   |
+--------------------------------------------------------------+
```

If the full-service resolver's cache have enough information to validate the query, the full-service resolver MAY use NSEC/NSEC3/

wildcard records aggressively. Otherwise, the full-service resolver MUST fall back to send the query to the authoritative DNS servers.

If the query name has the matching NSEC/NSEC3 RR and it proves the information requested does not exist, the full-service resolver may respond with a NODATA (empty) answer.

5.2. NSEC

If a full-service resolver implementation supports aggressive negative caching, then it SHOULD support aggressive use of NSEC and enable it by default. It SHOULD provide a configuration switch to disable aggressive use of NSEC and allow it to be enabled or disabled for specific zones.

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 the full-service resolver's cache contains an NSEC RR covering the source of synthesis and the covering NSEC RR of the query name, the full-service resolver may respond with NXDOMAIN error immediately.

5.3. NSEC3

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

If the full-service resolver's cache contains an NSEC3 RR matching the closest encloser, an NSEC3 RR covering the next closer name, and an NSEC3 RR covering the source of synthesis, it is possible for the full-service resolver to respond with NXDOMAIN immediately.

If a covering NSEC3 RR has 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.

A full-service resolver implementation MAY support aggressive use of NSEC3. It SHOULD provide a configuration switch to disable aggressive use of NSEC3 and allow it to be enabled or disabled for specific zones.

5.4. Wildcard

The last paragraph of RFC 4035 Section 4.5 discusses aggressive use of a cached deduced wildcard (as well as aggressive use of NSEC) and recommends that it is not relied upon.

Just like the case for the aggressive use of NSEC discussed in this draft, we could revisit this recommendation. As long as the full-service resolver knows a name would not exist without the wildcard match, it could answer a query for that name using the cached deduced wildcard, and it may be justified for performance and other benefits. (Note that, so far, this is orthogonal to "when aggressive use (of NSEC) is enabled").

Furthermore, when aggressive use of NSEC is enabled, the aggressive use of cached deduced wildcard will be more effective.

A full-service resolver implementation MAY support aggressive use of wildcards. It SHOULD provide a configuration switch to disable aggressive use of wildcards.

5.5. 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 RFC 2308 states the maximum number of negative cache TTL value is 3 hours (10800). So the full-service resolver SHOULD limit the maximum effective TTL value of negative responses (NSEC/NSEC3 RRs) to 10800 (3 hours). It is reasonably small but still effective for the purpose of this document, since it can eliminate significant amount of DNS attacks with randomly generated names.

6. Effects

6.1. Decrease of root DNS server queries

Aggressive use of NSEC/NSEC3 resource records results in a decrease of queries to the root - this decreases load on the root servers (the majority of queries currently result in NXDOMAIN responses), and increases privacy.

People may generate many typos in TLD, and they will result in unnecessary DNS queries. Some implementations leak non-existent TLD queries whose second level domain are different each other. Well observed examples are ".local" and ".belkin". With this proposal, it is possible to return NXDOMAIN immediately to such queries without further DNS recursive resolution process. It may reduces round trip

time, as well as reduces the DNS queries to corresponding authoritative servers, including Root DNS servers.

6.2. Mitigation of random subdomain attacks

Random sub-domain attacks (referred to as "Water Torture" attacks or NXDomain attacks) send many queries for non-existent information to full-service resolvers. Their query names consist of random prefixes and a target domain name. The negative cache does not work well, and thus targeted full-service resolvers end up sending queries to authoritative DNS servers of the target domain name.

When the number of queries is large, the full-service resolvers drop queries from both legitimate users and attackers as their outstanding queues are filled up.

For example, BIND 9.10.2 [BIND9] full-service resolvers answer SERVFAIL and Unbound 1.5.2 full-service resolvers drop most of queries under 10,000 queries per second attack.

The countermeasures implemented at this moment are rate limiting and disabling name resolution of target domain names in ad-hoc manner.

If the full-service resolver supports aggressive negative caching and the target domain name is signed with NSEC/NSEC3 (without Opt-Out), it may be used as a possible countermeasure of random subdomain attacks.

However, attackers can set the CD bit to their attack queries. The CD bit disables signature validation and the aggressive negative caching will be of no use.

7. Additional proposals

There are additional proposals to the aggressive negative caching.

7.1. Partial implementation

It is possible to implement aggressive negative caching partially.

DLV aggressive negative caching [RFC5074] is an implementation of NSEC aggressive negative caching which targets DLV domain names.

NSEC3 is somewhat more complex to implement, and some implementations may choose to only implement aggressive negative caching for NSEC.

Root only aggressive negative caching is also possible. It uses NSEC and RRSIG resource records whose signer domain name is root.

[I-D.wkumari-dnsop-cheese-shop] proposed root only aggressive negative caching in order to decrease defects and standardize quickly. The root zone has certain properties that make it a special case: It is DNSSEC signed and uses NSEC, the majority of the queries are "junk" queries, the rate of change is relatively slow, and there are no corner cases such as wildcards. Because of these properties, we know that generated negative answers will work.

7.2. Aggressive negative caching flag idea

Authoritative DNS servers that dynamically generate NSEC records normally generate minimally covering NSEC Records [RFC4470]. Aggressive negative caching does not work with minimally covering NSEC records. Most of DNS operators don't want zone enumeration and zone information leaks. They prefer NSEC resource records with narrow ranges. When a flag shows a full-service resolver supporting the aggressive negative caching and a query has the aggressive negative caching flag, authoritative DNS servers can generate NSEC resource records with wider range under random subdomain attacks. However, anyone (including attackers) can always use the flag..

8. IANA Considerations

This document has no IANA actions.

9. Security Considerations

Newly registered resource records may not be used immediately. However, choosing suitable TTL value 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 (3hrs), to mitigate this issue. Implementations which comply with this proposal are recommended to have a configurable maximum value of NSEC RRs in the negative cache.

Aggressive use of NSEC / NSEC3 resource records without DNSSEC validation may cause security problems. It is highly recommended to apply DNSSEC validation.

10. Implementation Status

Unbound has aggressive negative caching code in its DLV validator. The author implemented NSEC aggressive caching using Unbound and its DLV validator code.

11. Acknowledgments

The authors gratefully acknowledge DLV [RFC5074] author Samuel Weiler and Unbound developers. Olafur Gudmundsson and Pieter Lexis proposed aggressive negative caching flag idea. Valuable comments were provided by Bob Harold, Tatuya JINMEI, Shumon Huque, Mark Andrews, Casey Deccio, Bob Harold, Stephane Bortzmeyer and Matthijs Mekking.

12. Change History

This section is used for tracking the update of this document. Will be removed after finalize.

From draft-fujiwara-dnsop-nsec-aggressiveuse-03 -> draft-ietf-dnsop-nsec-aggressiveuse

- o Document adopted by DNSOP WG.
- o Adoption comments
- o Changed main purpose to performance
- o Use NSEC3/Wildcard keywords
- o Improved wordings (from good comments)
- o Simplified pseudo code for NSEC3
- o Added Warren as co-author.
- o Reworded much of the problem statement
- o Reworked examples to better explain the problem / solution.

12.1. Version draft-fujiwara-dnsop-nsec-aggressiveuse-01

- o Added reference to DLV [RFC5074] and imported some sentences.
- o Added Aggressive Negative Caching Flag idea.
- o Added detailed algorithms.

12.2. Version draft-fujiwara-dnsop-nsec-aggressiveuse-02

- o Added reference to [I-D.vixie-dnsext-resimprove]
- o Added considerations for the CD bit

- Updated detailed algorithms.
- Moved Aggressive Negative Caching Flag idea into Additional Proposals

12.3. Version draft-fujiwara-dnsop-nsec-aggressiveuse-03

- Added "Partial implementation"
- Section 4,5,6 reorganized for better representation
- Added NODATA answer in Section 4
- Trivial updates
- Updated pseudo code

13. References

13.1. 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>.

[RFC4470] Weiler, S. and J. Ihren, "Minimally Covering NSEC Records and DNSSEC On-line Signing", RFC 4470, DOI 10.17487/RFC4470, April 2006, <http://www.rfc-editor.org/info/rfc4470>.

[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>.

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

[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>.

[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>.

13.2. Informative References

[BIND9] Internet Systems Consortium, Inc., "Name Server Software", 2000, <https://www.isc.org/downloads/bind/>.

[I-D.ietf-dnsop-nxdomain-cut] Bortzmeyer, S. and S. Huque, "NXDOMAIN really means there is nothing underneath", draft-ietf-dnsop-nxdomain-cut-03 (work in progress), May 2016.

[I-D.vixie-dnsext-resimprove] Vixie, P., Joffe, R., and F. Neves, "Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness", draft-vixie-dnsext-resimprove-00 (work in progress), June 2010.

[I-D.wkumari-dnsop-cheese-shop] Kumari, W. and G. Huston, "Believing NSEC records in the DNS root.", draft-wkumari-dnsop-cheese-shop-01 (work in progress), February 2016.

[UNBOUND] NLnet Labs, "Unbound DNS validating resolver", 2006, <http://www.unbound.net/>.

Appendix A. Aggressive negative caching from RFC 5074

Imported from Section 6 of [RFC5074].

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 whenever it has an applicable validated NSEC in its cache unless the CD bit was set on the incoming query.

Imported from Section 6.1 of [RFC5074].

Implementing aggressive negative caching suggests that a validator will need to build an ordered data structure of NSEC records in order to efficiently find covering NSEC records. Only NSEC records from DLV domains need to be included in this data structure.

Appendix B. Detailed implementation idea

Section 6.1 of [RFC5074] is expanded as follows.

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.

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

If errors happen in aggressive negative caching algorithm, resolvers MUST fall back to resolve the query as usual. "Resolve the query as usual" means that the full-resolver resolve the query in Recursive-mode as if the full-service resolver does not implement aggressive negative caching.

To implement aggressive negative caching, resolver algorithm near cache lookup will be changed as follows:

```
QNAME = the query name;
QTYPE = the query type;
if ({QNAME,QTYPE} entry exists in the cache) {
    // the resolver responds the RRSet from the cache
    resolve the query as usual;
}

// if NSEC* exists, QTYPE existence is proved by type bitmap
if (matching NSEC/NSEC3 of QNAME exists in the cache) {
    if (QTYPE exists in type bitmap of NSEC/NSEC3 of QNAME) {
        // the entry exists, however, it is not in the cache.
        // need to iterate QNAME/QTYPE.
        resolve the query as usual;
    } else {
        // QNAME exists, QTYPE does not exist.
        the resolver can generate NODATA response;
```

```
       }
   }

   // Find closest enclosing NS RRset in the cache.
   // The owner of this NS RRset will be a suffix of the QNAME
   //    - the longest suffix of any NS RRset in the cache.
   SIGNER = closest enclosing NS RRSet of QNAME in the cache;

   // Check the NS RR of the SIGNER
   if (NS RR of SIGNER and its RRSIG RR do not exist in the cache
       or SIGNER zone is not signed or not validated) {
      Resolve the query as usual;
   }

   if (SIGNER zone does not have NSEC_TABLE) {
       Resolve the query as usual;
   }

   if (SIGNER zone is signed with NSEC) { // NSEC mode

       // Check the non-existence of QNAME
       CoveringNSEC = Find the covering NSEC of QNAME from NSEC_TABLE;
       if (Covering NSEC doesn't exist in the cache and NSEC_TABLE) {
           Resolve the query as usual.
       }

       // Select the longest existing name of QNAME from covering NSEC
       ClosestEncloser = common part of both owner name and
                           next domain name of CoveringNSEC;

       if (*.LongestExistName entry exists in the cache) {
           the resolver can generate positive response
           // synthesize the wildcard *.TEST
       }
       if covering NSEC RR of "*.LongestExistName" at SIGNER zone exists
            in the cache {
           the resolver can generate negative response;
       }
       //*.LongestExistName may exist. cannot generate negative response
       Resolve the query as usual.

   } else
   if (SIGNER zone is signed with NSEC3) {
       // NSEC3 mode

       ClosestEncloser = Find the closest encloser of QNAME
                                             from the cache
       // to prove the non-existence of QNAME,
```

```
       // closest encloser of QNAME must be in the cache

       NextCloserName = the next closer name of QNAME
       SourceOfSyhthesis = *.ClosestEncloser

       if (matching NSEC3 of ClosestEncloser exists in the cache
           and
           covering NSEC3 of NextCloserName exists in the cache
           and covering NSEC3 is not Opt-Out flag set) {

           // ClosestEncloser exists, and NextCloserName does not exist
           // then we need to check *.ClosestEncloser

           if (*.ClosestEncloser entry exists in the cache) {
               if (*.ClosestEncloser/QTYPE entry exists in the cache) {
                   the resolver can generate positive response
               } else {
                   // lack of *.ClosestEncloser/QTYPE information
                   Resolve the query as usual
               }
           } else
           if (covering NSEC3 of *.ClosestEncloser exists
               and covering NSEC3 is not Opt-Out flag set) {
               the resolver can generate negative response;
           }
       }
       // no matching/covering NSEC3 of QNAME information
       Resolve the query as usual
   }
```

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
US

Email: warren@kumari.net