

Domain Name System Operations (dnsop) Working Group  
Internet-Draft  
Updates: [1034](#), [2308](#) (if approved)  
Intended status: Standards Track  
Expires: March 22, 2017

S. Bortzmeyer  
AFNIC  
S. Huque  
Verisign Labs  
September 18, 2016

**NXDOMAIN really means there is nothing underneath  
draft-ietf-dnsop-nxdomain-cut-05**

Abstract

This document states clearly that when a DNS resolver receives a response with response code of NXDOMAIN, it means that the domain name which is thus denied AND ALL THE NAMES UNDER IT do not exist.

REMOVE BEFORE PUBLICATION: this document should be discussed in the IETF DNSOP (DNS Operations) group, through its mailing list. The source of the document, as well as a list of open issues, is currently kept at Github [[1](#)].

This documents clarifies [RFC 1034](#) and modifies a portion of [RFC 2308](#), so it updates both of them.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 22, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction and background . . . . .	<a href="#">2</a>
<a href="#">1.1.</a>	Terminology . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Rule . . . . .	<a href="#">3</a>
<a href="#">3.</a>	Updates to RFCs . . . . .	<a href="#">5</a>
<a href="#">3.1.</a>	Updates to <a href="#">RFC1034</a> . . . . .	<a href="#">5</a>
<a href="#">3.2.</a>	Updates to <a href="#">RFC2308</a> . . . . .	<a href="#">5</a>
<a href="#">4.</a>	Benefits . . . . .	<a href="#">5</a>
<a href="#">5.</a>	Possible issues . . . . .	<a href="#">6</a>
<a href="#">6.</a>	Implementation considerations . . . . .	<a href="#">6</a>
<a href="#">7.</a>	IANA Considerations . . . . .	<a href="#">7</a>
<a href="#">8.</a>	Security Considerations . . . . .	<a href="#">7</a>
<a href="#">9.</a>	Implementation status - RFC EDITOR: REMOVE BEFORE PUBLICATION	<a href="#">7</a>
<a href="#">10.</a>	Acknowledgments . . . . .	<a href="#">8</a>
<a href="#">11.</a>	References . . . . .	<a href="#">8</a>
<a href="#">11.1.</a>	Normative References . . . . .	<a href="#">8</a>
<a href="#">11.2.</a>	Informative References . . . . .	<a href="#">9</a>
<a href="#">11.3.</a>	URIs . . . . .	<a href="#">10</a>
<a href="#">Appendix A.</a>	Why can't we just use the owner name of the returned SOA? . . . . .	<a href="#">10</a>
<a href="#">Appendix B.</a>	Related approaches . . . . .	<a href="#">11</a>
	Authors' Addresses . . . . .	<a href="#">11</a>

## [1.](#) Introduction and background

The DNS protocol [[RFC1035](#)] defines response code 3 as "Name Error", or "NXDOMAIN" [[RFC2308](#)], which means that the queried domain name does not exist in the DNS. Since domain names are represented as a tree of labels ([RFC1034](#), [Section 3.1](#)), non-existence of a node implies non-existence of the entire sub-tree rooted at this node.

The DNS iterative resolution algorithm precisely interprets the NXDOMAIN signal in this manner. If it encounters an NXDOMAIN response code from an authoritative server, it immediately stops iteration and returns the NXDOMAIN response to the querier.

However, in most known existing resolvers today, a cached non-existence for a domain is not considered "proof" that there can be no child domains underneath. This is due to an ambiguity in [[RFC1034](#)]



that failed to distinguish Empty Non-Terminal names (ENT) ([\[RFC7719\]](#)) from nonexistent names ([Section 3.1](#)). The distinction became especially important for the development of DNSSEC, which provides proof of non-existence. [\[RFC4035\]](#), [section 3.1.3.2](#), describes how security-aware authoritative name servers make the distinction, but no existing RFCs describe the behavior for recursive name servers.

This document specifies that an NXDOMAIN response for a domain name means that no child domains underneath the queried name exist either. And furthermore, that DNS resolvers should interpret cached non-existence in this manner. Since the domain names are organized in a tree, it is a simple consequence of the tree structure: non-existence of a node implies non-existence of the entire sub-tree rooted at this node.

## 1.1. 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 [\[RFC2119\]](#).

"Denied name": the domain name whose existence has been denied by a response of rcode NXDOMAIN. In most cases, it is the QNAME but, because of [\[RFC6604\]](#), it is not always the case.

Other terms are defined in [\[RFC1034\]](#) or [\[RFC1035\]](#) or (like NXDOMAIN itself) in the more recent [\[RFC7719\]](#).

The domain name space is conceptually defined in terms of a tree structure. The implementation of a DNS resolver/cache MAY use a tree or other data structures. The cache being a subset of the data in the domain name space, it is much easier to reason about it in terms of that tree structure and to describe things in those terms (names under/above, descendent names, subtrees etc). In fact, the DNS algorithm description in [\[RFC1034\]](#) even states an assumption that the cache is a tree structure, so the precedent is already well established: see its [section 4.3.2](#) which says "The following algorithm assumes that the RRs are organized in several tree structures, one for each zone, and another for the cache..." So, in this document, each time we talk of a tree or tree operations, it refers to the model, not to the actual implementation.

## 2. Rule

When an iterative caching DNS resolver receives a response NXDOMAIN, it SHOULD store it in its cache and all names and RRsets at or below that node SHOULD then be considered to be unreachable. Subsequent queries for such names SHOULD elicit an NXDOMAIN response.



But if a resolver has cached data under the NXDOMAIN cut, it MAY continue to send it as a reply (until the TTL of this cached data expires), since this may avoid additional processing when a query is received. [Section 6](#) provides more information about this.

Another exception is that a validating resolver MAY decide to implement the "NXDOMAIN cut" behaviour (described in the first paragraph of this section) only when the NXDOMAIN response has been validated with DNSSEC. See [Section 8](#) for the rationale.

The fact that a subtree does not exist is not forever: [\[RFC2308\], section 3](#), already describes the amount of time that an NXDOMAIN response may be cached (the "negative TTL").

If the NXDOMAIN response due to a cached non-existence is from a DNSSEC signed zone, then it will have accompanying NSEC or NSEC3 records that authenticate the non-existence of the name. For a descendant name of the original NXDOMAIN name, the same set of NSEC or NSEC3 records proves the non-existence of the descendant name. The iterative, caching resolver MUST return these NSEC or NSEC3 records in the response to the triggering query if the query had the DNSSEC OK (DO) bit set.

Warning: if there is a chain of CNAME (or DNAME), the name which does not exist is the last of the chain ([\[RFC6604\]](#)) and not the QNAME. The NXDOMAIN stored in the cache is for the denied name, not always for the QNAME.

As an example of the consequence of these rules, consider two successive queries to a resolver, with a non-existing domain 'foo.example': the first is for 'foo.example' (which results in an NXDOMAIN) and the second for 'bar.foo.example' (which also results in an NXDOMAIN). Many resolvers today will forward both queries, as noticed in [Section 9](#). However, following the rules in this document ("NXDOMAIN cut"), a resolver would cache the first NXDOMAIN response, as a sign of non-existence, and then immediately return an NXDOMAIN response for the second query, without transmitting it to an authoritative server.

If the first request is for 'bar.foo.example' and the second for 'baz.foo.example', the first NXDOMAIN response won't tell anything about 'baz.foo.example' and therefore the second query will be transmitted as it was before the use of "NXDOMAIN cut" optimisation (see [Appendix A](#)).



### 3. Updates to RFCs

#### 3.1. Updates to [RFC1034](#)

This document clarifies possible ambiguities in [\[RFC1034\]](#) that did not clearly distinguish Empty Non-Terminal names (ENT) ([\[RFC7719\]](#)) from nonexistent names, and refers to subsequent documents that do. Empty Non-Terminals are nodes in the DNS that have no resource record sets associated with them, but have descendant nodes that do. The correct response to Empty Non-Terminals is NODATA (i.e. a response code of NOERROR and an empty ANSWER section). Additional clarifying language on these points is provided in [section 7.16 of \[RFC2136\]](#) and sections [2.2.2](#) and [2.2.3](#) of [\[RFC4592\]](#).

#### 3.2. Updates to [RFC2308](#)

The second paragraph of [section 5 of \[RFC2308\]](#) states the following: "A negative answer that resulted from a name error (NXDOMAIN) should be cached such that it can be retrieved and returned in response to another query for the same <QNAME, QCLASS> that resulted in the cached negative response."

This document revises that paragraph to the following: "A negative answer that resulted from a name error (NXDOMAIN) should be cached such that it can be retrieved and returned in response to another query for the same <QNAME, QCLASS> that resulted in the cached negative response, or where QNAME is a descendant of the original QNAME, and QCLASS is the same."

The above section [Section 2](#) elaborates on the revised rule and specifies when it may be reasonable to relax or ignore it.

### 4. Benefits

The main benefit is a better efficiency of the caches. In the example above, the resolver sends only one query instead of two, the second one being answered from the cache. This will benefit the entire DNS ecosystem, since the authoritative name servers will have less unnecessary traffic to process.

The correct behavior (in [\[RFC1034\]](#) and made clearer in this document) is specially useful when combined with QNAME minimisation [\[RFC7816\]](#) since it will allow a resolver to stop searching as soon as an NXDOMAIN is encountered.

"NXDOMAIN cut" may also help mitigate certain types of random QNAME attacks [\[joost-dnsterior\]](#) [\[balakrichen-dafa888\]](#), where there is a fixed suffix which does not exist. In these attacks against the





authoritative name server, queries are sent to resolvers for a QNAME composed of a fixed suffix ("dafa888.wf" in one of the articles above), which is typically nonexistent, and a random prefix, different for each request. A resolver receiving these requests have to forward them to the authoritative servers. With "NXDOMAIN cut", a system administrator would just have to send to the resolver a query for the fixed suffix, the resolver would get a NXDOMAIN and then would stop forwarding the queries. (It would be better if the SOA record in the NXDOMAIN response were sufficient to find the non-existing domain but it is not the case, see [Appendix A](#).)

## 5. Possible issues

Let's assume the TLD example exists but foobar.example is not delegated (so the example's name servers will reply NXDOMAIN for a query about anything.foobar.example). A system administrator decides to name the internal machines of his organization under office.foobar.example and uses a trick of his resolver to forward requests about this zone to his local authoritative name servers. "NXDOMAIN cut" would create problems here, since, depending on the order of requests to the resolver, it may have cached the non-existence from example and therefore "deleted" everything under. This document assumes that such setup is rare and does not need to be supported.

Another issue that may happen: today, we see broken authoritative name servers which reply to ENT ([\[RFC7719\], section 6](#)) with NXDOMAIN instead of the normal NODATA ([\[RFC7719\], section 3](#)).

RFC-EDITOR: REMOVE THE PARAGRAPH BEFORE PUBLICATION. An example today is mta2.\_domainkey.cbs.nl (which exists) where querying \_domainkey.cbs.nl yields NXDOMAIN. Another example is www.upenn.edu, redirected to www.upenn.edu-dscg.edgesuite.net while a query for edu-dscg.edgesuite.net returns NXDOMAIN.

Such name servers are definitely wrong and have always been. Their behaviour is incompatible with DNSSEC. Given the advantages of "NXDOMAIN cut", there is little reason to support this behavior.

## 6. Implementation considerations

This section is non-normative, and is composed only of various things which may be useful for implementors. A recursive resolver may implement its cache in many ways. The most obvious one is a tree data structure, because it fits the data model of domain names. But, in practice, other implementations are possible, as well as various optimisations (such as a tree, augmented by an index of some common domain names).



If a resolver implements its cache as a tree (without any optimisation), one way to follow the rules of [Section 2](#) is, when receiving the NXDOMAIN, to prune the subtree of positive cache entries at that node, or to delete all individual cache entries for names below that node. Then, when searching downward in its cache, this iterative caching DNS resolver will stop searching if it encounters a cached non-existence.

Some resolver may have a cache which is NOT organized as a tree (but, for instance, as a dictionary) and therefore have a reason to ignore the rules of [Section 2](#). So these rules are a SHOULD and not a MUST.

## **7. IANA Considerations**

This document has no actions for IANA.

## **8. Security Considerations**

The technique described here may help against a denial-of-service attack named "random qnames" and described in [Section 4](#).

If a resolver does not validate the answers with DNSSEC, or if the zone is not signed, the resolver can of course be poisoned with a false NXDOMAIN, thus "deleting" a part of the domain name tree. This denial-of-service attack is already possible without the rules of this document (but "NXDOMAIN cut" may increase its effects). The only solution is to use DNSSEC.

## **9. Implementation status - RFC EDITOR: REMOVE BEFORE PUBLICATION**

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [\[RFC6982\]](#). The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [\[RFC6982\]](#), "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature.



It is up to the individual working groups to use this information as they see fit".

As of today, practically all existing DNS resolvers are conservative by default: they consider a NXDOMAIN as only significant for the denied name itself, not for the names under. Almost all the current recursive servers will send upstream a query for out-of-cache sub.example.com even if their cache contains an NXDOMAIN for example.com.

There are a few exceptions. The Unbound resolver has a configuration parameter called "harden-below-nxdomain" [2], which if set to "yes" turns on "NXDOMAIN cut" behavior ("only DNSSEC-secure nxdomains are used", see [Section 8](#)). The PowerDNS recursor has optional partial support for "NXDOMAIN cut", for the root domain only, with its "root-nx-trust" setting, described as [3] "If set, an NXDOMAIN from the root-servers will serve as a blanket NXDOMAIN for the entire TLD the query belonged to. The effect of this is far fewer queries to the root-servers.".

## **10. Acknowledgments**

The main idea in this document is taken from [[I-D.vixie-dnsext-resimprove](#)], section 3, "Stopping Downward Cache Search on NXDOMAIN". Thanks to its authors, Paul Vixie, Rodney Joffe, and Frederico Neves. Additionally Tony Finch, Ted Lemon, John Levine, Jinmei Tatuya, Bob Harold and Duane Wessels provided valuable feedback and suggestions.

## **11. References**

### **11.1. Normative References**

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, [RFC 1034](#), DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, [RFC 1035](#), DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [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>>.



- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", [RFC 2136](#), DOI 10.17487/RFC2136, April 1997, <<http://www.rfc-editor.org/info/rfc2136>>.
- [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>>.
- [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>>.
- [RFC6604] Eastlake 3rd, D., "xNAME RCODE and Status Bits Clarification", [RFC 6604](#), DOI 10.17487/RFC6604, April 2012, <<http://www.rfc-editor.org/info/rfc6604>>.

## **11.2. Informative References**

- [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>>.
- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", [RFC 6982](#), DOI 10.17487/RFC6982, July 2013, <<http://www.rfc-editor.org/info/rfc6982>>.
- [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>>.
- [RFC7816] Bortzmeyer, S., "DNS Query Name Minimisation to Improve Privacy", [RFC 7816](#), DOI 10.17487/RFC7816, March 2016, <<http://www.rfc-editor.org/info/rfc7816>>.
- [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.ietf-dnsop-nsec-aggressiveuse] Fujiwara, K., Kato, A., and W. Kumari, "Aggressive use of NSEC/NSEC3", [draft-ietf-dnsop-nsec-aggressiveuse-02](#) (work in progress), September 2016.





[joost-dnsterror]

Joost, M., "About DNS Attacks and ICMP Destination Unreachable Reports", December 2014, <<http://www.michael-joost.de/dnsterror.html>>.

[balakrichenan-dafa888]

Balakrichenan, S., "Disturbance in the DNS - "Random qnames", the dafa888 DoS attack", October 2014, <<https://indico.dns-oarc.net/event/20/session/3/contribution/37>>.

### **11.3. URIs**

- [1] <https://github.com/bortzmeyer/ietf-dnsop-nxdomain>
- [2] <https://www.unbound.net/documentation/unbound.conf.html>
- [3] <https://doc.powerdns.com/md/recursor/settings/#root-nx-trust>

### **Appendix A. Why can't we just use the owner name of the returned SOA?**

In this document, we deduce the non-existence of a domain only for NXDOMAIN answers where the denied name was the exact domain. If a resolver sends a query to the name servers of the TLD example, asking for the MX record for `www.foobar.example`, and subsequently receives a NXDOMAIN, it can only register the fact that `www.foobar.example` (and everything underneath) does not exist. This is true regardless if the accompanying SOA record is for the domain `example` only. One cannot infer that `foobar.example` is nonexistent. The accompanying SOA record indicates the apex of the zone, not the closest existing domain name. So, using the owner name of the SOA record in the Authoritative section to deduce "NXDOMAIN cuts" is currently definitely not OK.

RFC-EDITOR: REMOVE BEFORE PUBLICATION: to use a real example today, ask the authoritative name servers of the TLD `fr` about `anything.which.does.not.exist.gouv.fr`. The SOA will indicate `fr` (the apex) even while `gouv.fr` does exist (there is no zone cut between `gouv.fr` and `fr`).

Deducing the non-existence of a node from the SOA in the NXDOMAIN reply may certainly help with random qnames attacks but this is out-of-scope for this document. It would require to address the problems mentioned in the first paragraph of this section. A possible solution would be, when receiving a NXDOMAIN with a SOA which is more than one label up in the tree, to send requests for the domains which are between the QNAME and the owner name of the SOA. (A resolver



which does DNSSEC validation or QNAME minimisation will need to do it, anyway.)

## **Appendix B. Related approaches**

The document [[I-D.ietf-dnsop-nsec-aggressiveuse](#)] describes another way to address some of the same concerns (decreasing the traffic for non-existing domain names). Unlike "NXDOMAIN cut", it requires DNSSEC but it is more powerful since it can synthesize NXDOMAINs for domains that were not queried.

### Authors' Addresses

Stephane Bortzmeyer  
AFNIC  
1, rue Stephenson  
Montigny-le-Bretonneux 78180  
France

Phone: +33 1 39 30 83 46  
Email: [bortzmeyer+ietf@nic.fr](mailto:bortzmeyer+ietf@nic.fr)  
URI: <http://www.afnic.fr/>

Shumon Huque  
Verisign Labs  
12061 Bluemont Way  
Reston 20190  
USA

Email: [shuque@verisign.com](mailto:shuque@verisign.com)  
URI: <http://www.verisignlabs.com/>