NXDOMAIN really means there is nothing underneath

draft-ietf-dnsop-nxdomain-cut-00

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 bit 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 June 29, 2016.

Copyright Notice

Copyright (c) 2015 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
1. Introduction and background

The DNS protocol [RFC1035] defines response code 3 as "Name Error", or "NXDOMAIN", i.e. 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 virtually all existing resolvers, a cached NXDOMAIN 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. For DNSSEC, the IETF had to distinguish this case ([RFC4035], section 3.1.3.2), but the implication on non-DNSSEC resolvers wasn't fully realized.
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 NXDOMAIN responses 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.

### 2. Rules

When searching downward in its cache, an iterative caching DNS resolver **SHOULD** stop searching if it encounters a cached NXDOMAIN. The response to the triggering query should be NXDOMAIN.

When an iterative caching DNS resolver stores an NXDOMAIN in its cache, all names and RRsets at or below that node **SHOULD** be considered to be unreachable, and subsequent queries for such names **SHOULD** elicit an NXDOMAIN response. A resolver, depending on its implementation, **MAY** prune the subtree of positive cache entries at that node, or delete all individual cache entries for names below that node. [TODO: currently under discussion, some people find it dangerous. Only if the NXDOMAIN is DNSSEC-validated? Perhaps the resolver could be configured to not "bulk delete" TLDs (or root). See [Section 6].]

By implication, a stream of queries foo.example, then bar.foo.example, where foo.example does not exist would normally cause both queries to be forwarded to example's nameservers. Following this recommended practice of "NXDOMAIN cut", the second query and indeed any other query for names at or below foo.example would not be forwarded.

These rules replace the second paragraph of [section 5 of RFC2308]. Otherwise, [RFC2308] applies unchanged, and the fact that a subtree does not exist is not forever: the NXDOMAIN is cached only for the duration of the "negative TTL" ([section 3 of RFC2308]).
If the cached NXDOMAIN response 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.

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.

3. Benefits

The main benefit is a better efficiency of the caches. In the example above, we send only one query instead of two, the second one being answered from the cache.

The correct behavior (in [RFC1034] and made clearer in this document) is especially useful when combined with QNAME minimisation [I-D.ietf-dnsop-qname-minimisation] 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-dnsterror] [balakrichenan-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, we 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.)

Since the principles set in this document are so great, why are the rules of Section 2 SHOULD and not MUST? This is because some resolver may have a cache which is NOT organized as a tree (but, for instance, as a dictionary) and therefore have a good reason to ignore this.
4. 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 use 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 NXDOMAIN 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. Given the advantages of NXDOMAIN cuts, there is little reason to support this behavior.

5. IANA Considerations

This document has no actions for IANA.

6. Security Considerations

The technique described here may help against a denial-of-service attack named "random qnames" and described in Section 3. Apart from that, it is believed to have no security consequences.

If a resolver does not validate the answers with DNSSEC, it 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.

7. 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: they consider a NXDOMAIN as only significant for the 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 6). 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.".

8. Acknowledgments

The main idea is 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, John Levine, Jinmei Tatuya, and Duane Wessels provided valuable feedback and suggestions.

9. References
9.1. Normative References


9.2. Informative References


[I-D.ietf-dnsop-qname-minimisation]
Bortzmeyer, S., "DNS query name minimisation to improve privacy", draft-ietf-dnsop-qname-minimisation-08 (work in progress), November 2015.

[I-D.fujiwara-dnsop-nsec-aggressiveuse]

[joost-dnsterror]

[balakrichenan-dafa888]

9.3. URIs


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 this exact domain. If a resolver sends a query to the name servers of the TLD example, and asks the MX record for www.foobar.example, and receives a NXDOMAIN, it can only register the fact that www.foobar.example (and everything underneath) does not exist. Even if the accompanying SOA record is for example only, one cannot infer that foobar.example is nonexistent. The accompanying SOA indicates the apex of the zone, not the closest existing domain name.

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.fujiwara-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 synthetize 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  

**Shumon Huque**  
Verisign labs  
12061 Bluemont Way  
Reston 20190  
USA  
Email: shuque@verisign.com  