Internet Engineering Task Force (IETF)
Request for Comments: 8020
Updates: 1034, 2308
Category: Standards Track
ISSN: 2070-1721

S. Bortzmeyer
AFNIC
S. Huque
Verisign Labs

November 2016

NXDOMAIN: There Really Is Nothing Underneath

Abstract

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

This document clarifies RFC 1034 and modifies a portion of RFC 2308: it updates both of them.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc8020.

Table of Contents

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), nonexistence of a node implies nonexistence of the entire subtree 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 nonexistence 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 (ENT) names ([RFC7719]) from nonexistent names (Section 3.1). The distinction became especially important for the development of DNSSEC, which provides proof of nonexistence. [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; furthermore, it means that DNS resolvers should interpret cached nonexistence in this manner. Since the domain names are organized in a tree, it is a simple consequence of the tree structure: nonexistence of a node implies nonexistence of the entire subtree 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].

"QNAME": defined in [RFC1034] and in [RFC1035], Section 4.1.2, but, because [RFC2308] provides a different definition, we repeat the original one here: the QNAME is the domain name in the question section.

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

Other terms are defined in [RFC1034], [RFC1035], and (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, descendant 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 about a tree or tree operations, we're referring to the model, not to the actual implementation.

2. Rules

When an iterative caching DNS resolver receives an NXDOMAIN response, it SHOULD store it in its cache and then all names and resource record sets (RRsets) at or below that node SHOULD be considered 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" behavior (described in the first paragraph of this section) only when the NXDOMAIN response has been validated with DNSSEC. See Section 7 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 nonexistence is from a DNSSEC-signed zone, then it will have accompanying NSEC or NSEC3 records that authenticate the nonexistence of the name. For a descendant name of the original NXDOMAIN name, the same set of NSEC or NSEC3 records proves the nonexistence 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 that 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 nonexisting 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. However, following the rules in this document ("NXDOMAIN cut"), a resolver would cache the first NXDOMAIN response, as a sign of nonexistence, 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', then the first NXDOMAIN response won't tell anything about 'baz.foo.example'; therefore, the second query will be transmitted as it was before the use of "NXDOMAIN cut" optimization (see Appendix A).

3. Updates to RFCs

3.1. Updates to RFC 1034

This document clarifies possible ambiguities in [RFC1034] that did not clearly distinguish Empty Non-Terminal (ENT) names ([RFC7719]) from nonexistent names, and it refers to subsequent documents that do. ENTs are nodes in the DNS that do not have resource record sets associated with them but have descendant nodes that do. The correct response to ENTs 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 in Sections 2.2.2 and 2.2.3 of [RFC4592].

3.2. Updates to RFC 2308

The second paragraph of Section 5 in [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 the QNAME is a descendant of the original QNAME and the QCLASS is the same.

Section 2 above 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 especially useful when combined with QNAME minimization [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-dnsterror] and [balakrichenan-dafa888], where there is a fixed suffix that 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 has 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 nonexisting domain, but this is not the case, see Appendix A.)

5. Possible Issues

Let's assume that the Top-Level Domain (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; depending on the order of requests to the resolver, it may have cached the nonexistence from example and therefore "deleted" everything under it. This document assumes that such a setup is rare and does not need to be supported.

Today, another possible issue exists; we see authoritative name servers that reply to ENT ([RFC7719], Section 6) with NXDOMAIN instead of the normal NODATA ([RFC7719], Section 3).

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 that 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 optimizations (such as a tree, augmented by an index of some common domain names).

If a resolver implements its cache as a tree (without any optimization), one way to follow the rules in Section 2 is as follows: when receiving the NXDOMAIN, prune the subtree of positive cache entries at that node or 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 nonexistence.

Some resolvers may have a cache that is NOT organized as a tree (but, for instance, as a dictionary); therefore, they have a reason to ignore the rules of Section 2. So these rules use SHOULD and not MUST.

7. Security Considerations

The technique described in this document may help against a denial-of-service attack named "random qnames" 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.

8. References

8.1. 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>.

8.2. 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>.

[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>.

[DNSRRR] 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.

[NSEC] Fujiwara, K., Kato, A., and W. Kumari, "Aggressive use of NSEC/NSEC3", Work in Progress, draft-ietf-dnsop-nsec-aggressiveuse-04, 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/3>.

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

In this document, we deduce the nonexistence 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 mail exchange (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 of whether or not 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 authority section to deduce "NXDOMAIN cuts" is currently definitely not OK.

Deducing the nonexistence 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 addressing the problems mentioned in the first paragraph of this section. A possible solution is, when receiving a NXDOMAIN with a SOA that is more than one label up in the tree, to send requests for the domains that are between the QNAME and the owner name of the SOA. (A resolver that does DNSSEC validation or QNAME minimization will need to do it anyway.)

Appendix B. Related Approaches

The document [NSEC] describes another way to address some of the same concerns (decreasing the traffic for nonexisting domain names). Unlike "NXDOMAIN cut", it requires DNSSEC, but it is more powerful since it can synthesize NXDOMAINs for domains that were not queried.

Acknowledgments

The main idea in this document is taken from [DNSRRR], 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.

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
URI: https://www.afnic.fr/

Shumon Huque
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
United States of America

Email: shuque@verisign.com
URI: http://www.verisignlabs.com/