Stream: Internet Engineering Task Force (IETF)
RFC: 9156
Obsoletes: 7816
Category: Standards Track
Published: November 2021
ISSN: 2070-1721
Authors: S. Bortzmeyer, *AFNIC*; R. Dolmans, *NLnet Labs*; P. Hoffman, *ICANN*

RFC 9156
DNS Query Name Minimisation to Improve Privacy

Abstract

This document describes a technique called "QNAME minimisation" to improve DNS privacy, where the DNS resolver no longer always sends the full original QNAME and original QTYPE to the upstream name server. This document obsoletes RFC 7816.

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 https://www.rfc-editor.org/info/rfc9156.

Table of Contents

1. Introduction and Background

The problem statement for this document is described in [RFC9076]. This specific solution is not intended to fully solve the DNS privacy problem; instead, it should be viewed as one tool amongst many.

QNAME minimisation follows the principle explained in Section 6.1 of [RFC6973]: the less data you send out, the fewer privacy problems you have.

Before QNAME minimisation, when a resolver received the query "What is the AAAA record for www.example.com?", it sent to the root (assuming a resolver, whose cache is empty) the very same question. Sending the full QNAME to the authoritative name server was a tradition, not a protocol requirement. In a conversation with one of the authors in January 2015, Paul

Mockapetris explained that this tradition comes from a desire to optimise the number of requests, when the same name server is authoritative for many zones in a given name (something that was more common in the old days, where the same name servers served .com and the root) or when the same name server is both recursive and authoritative (something that is strongly discouraged now). Whatever the merits of this choice at this time, the DNS is quite different now.

QNAME minimisation is compatible with the current DNS system and therefore can easily be deployed. Because it is only a change to the way that the resolver operates, it does not change the DNS protocol itself. The behaviour suggested here (minimising the amount of data sent in QNAMEs from the resolver) is allowed by Section 5.3.3 of [RFC1034] and Section 7.2 of [RFC1035].

1.1. Experience from RFC 7816

This document obsoletes [RFC7816]. [RFC7816] was categorised "Experimental", but ideas from it were widely deployed since its publication. Many resolver implementations now support QNAME minimisation. The lessons learned from implementing QNAME minimisation were used to create this new revision.

Data from DNSThought [dnsthought-qnamemin], Verisign [verisign-qnamemin], and APNIC [apnic-qnamemin] shows that a large percentage of the resolvers deployed on the Internet already support QNAME minimisation in some way.

Academic research has been performed on QNAME minimisation [devries-qnamemin]. This work shows that QNAME minimisation in relaxed mode causes almost no problems. The paper recommends using the A QTYPE and limiting the number of queries in some way. Some of the issues that the paper found are covered in Section 5.

1.2. Terminology

The terminology used in this document is defined in [RFC8499].

In this document, a "cold" cache is one that is empty, having literally no entries in it. A "warm" cache is one that has some entries in it.

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.

2. Description of QNAME Minimisation

The idea behind QNAME minimisation is to minimise the amount of privacy-sensitive data sent from the DNS resolver to the authoritative name server. This section describes how to do QNAME minimisation. The algorithm is summarised in Section 3.

When a resolver is not able to answer a query from cache, it has to send a query to an authoritative name server. Traditionally, these queries would contain the full QNAME and the original QTYPE as received in the client query.

The full QNAME and original QTYPE are only needed at the name server that is authoritative for the record requested by the client. All other name servers queried while resolving the query only need to receive enough of the QNAME to be able to answer with a delegation. The QTYPE in these queries is not relevant, as the name server is not able to authoritatively answer the records the client is looking for. Sending the full QNAME and original QTYPE to these name servers therefore exposes more privacy-sensitive data than necessary to resolve the client's request.

A resolver that implements QNAME minimisation obscures the QNAME and QTYPE in queries directed to an authoritative name server that is not known to be responsible for the original QNAME. These queries contain:

- a QTYPE selected by the resolver to possibly obscure the original QTYPE
- the QNAME that is the original QNAME, stripped to just one label more than the longest matching domain name for which the name server is known to be authoritative

2.1. QTYPE Selection

Note that this document relaxes the recommendation in [RFC7816] to use the NS QTYPE to hide the original QTYPE. Using the NS QTYPE is still allowed. The authority of NS records lies at the child side. The parent side of the delegation will answer using a referral, like it will do for queries with other QTYPEs. Using the NS QTYPE therefore has no added value over other QTYPEs.

The QTYPE to use while minimising queries can be any possible data type (as defined in Section 3.1 of [RFC6895]) for which the authority always lies below the zone cut (i.e., not DS, NSEC, NSEC3, OPT, TSIG, TKEY, ANY, MAILA, MAILB, AXFR, and IXFR), as long as there is no relation between the incoming QTYPE and the selection of the QTYPE to use while minimising. The A or AAAA QTYPEs are always good candidates to use because these are the least likely to raise issues in DNS software and middleboxes that do not properly support all QTYPEs. QTYPE=A or QTYPE=AAAA queries will also blend into traffic from nonminimising resolvers, making it in some cases harder to observe that the resolver is using QNAME minimisation. Using a QTYPE that occurs most in incoming queries will slightly reduce the number of queries, as there is no extra check needed for delegations on non-apex records.

2.2. QNAME Selection

The minimising resolver works perfectly when it knows the zone cut (zone cuts are described in Section 6 of [RFC2181]). But zone cuts do not necessarily exist at every label boundary. In the name www.foo.bar.example, it is possible that there is a zone cut between "foo" and "bar" but not between "bar" and "example". So, assuming that the resolver already knows the name servers of example, when it receives the query "What is the AAAA record of www.foo.bar.example?", it does not always know where the zone cut will be. To find the zone cut, it will query the example name servers for a record for bar.example. It will get a non-referral answer, so it has to query the example name servers again with one more label, and so on. (Section 3 describes this algorithm in deeper detail.)

2.3. Limitation of the Number of Queries

When using QNAME minimisation, the number of labels in the received QNAME can influence the number of queries sent from the resolver. This opens an attack vector and can decrease performance. Resolvers supporting QNAME minimisation **MUST** implement a mechanism to limit the number of outgoing queries per user request.

Take for example an incoming QNAME with many labels, like www.host.group.department.example.com, where host.group.department.example.com is hosted on example.com's name servers. (Such deep domains are especially common under ip6.arpa.) Assume a resolver that knows only the name servers of example.com. Without QNAME minimisation, it would send these example.com name servers a query for www.host.group.department.example.com and immediately get a specific referral or an answer, without the need for more queries to probe for the zone cut. For such a name, a cold resolver with QNAME minimisation will send more queries, one per label. Once the cache is warm, there will be less difference with a traditional resolver. Testing of this is described in [Huque-QNAME-Min].

The behaviour of sending multiple queries can be exploited by sending queries with a large number of labels in the QNAME that will be answered using a wildcard record. Take for example a record for *.example.com, hosted on example.com's name servers. An incoming query containing a QNAME with more than 100 labels, ending in example.com, will result in a query per label. By using random labels, the attacker can bypass the cache and always require the resolver to send many queries upstream. Note that [RFC8198] can limit this attack in some cases.

One mechanism that **MAY** be used to reduce this attack vector is by appending more than one label per iteration for QNAMEs with a large number of labels. To do this, a maximum number of QNAME minimisation iterations **MUST** be selected (MAX_MINIMISE_COUNT); a **RECOMMENDED** value is 10. Optionally, a value for the number of queries that should only have one label appended **MAY** be selected (MINIMISE_ONE_LAB); a good value is 4. The assumption here is that the number of labels on delegations higher in the hierarchy are rather small; therefore, not exposing too many labels early on has the most privacy benefit.

Another potential, optional mechanism for limiting the number of queries is to assume that labels that begin with an underscore (_) character do not represent privacy-relevant administrative boundaries. For example, if the QNAME is "_25._tcp.mail.example.org" and the algorithm has already searched for "mail.example.org", the next query can be for all the underscore-prefixed names together, namely "_25._tcp.mail.example.org".

When a resolver needs to send out a query, it will look for the closest-known delegation point in its cache. The number of not-yet-exposed labels is the difference between this closest name server and the incoming QNAME. The first MINIMISE_ONE_LAB labels will be handled as described in Section 2. The number of labels that are still not exposed now need to be divided proportionally over the remaining iterations (MAX_MINIMISE_COUNT - MINIMISE_ONE_LAB). If the not-yet-exposed labels cannot be equally divided over the remaining iterations, the remainder of the division should be added to the last iterations. For example, when resolving a QNAME with 18 labels with MAX_MINIMISE_COUNT set to 10 and MINIMISE_ONE_LAB set to 4, the number of labels added per iteration are: 1,1,1,1,2,2,2,2,3,3.

2.4. Implementation by Stub and Forwarding Resolvers

Stub and forwarding resolvers **MAY** implement QNAME minimisation. Minimising queries that will be sent to an upstream resolver does not help in hiding data from the upstream resolver because all information will end up there anyway. It might however limit the data exposure between the upstream resolver and the authoritative name server in the situation where the upstream resolver does not support QNAME minimisation. Using QNAME minimisation in a stub or forwarding resolver that does not have a mechanism to find and cache zone cuts will drastically increase the number of outgoing queries.

3. Algorithm to Perform QNAME Minimisation

This algorithm performs name resolution with QNAME minimisation in the presence of zone cuts that are not yet known.

Although a validating resolver already has the logic to find the zone cuts, implementers of resolvers may want to use this algorithm to locate the zone cuts.

(0) If the query can be answered from the cache, do so; otherwise, iterate as follows:

(1) Get the closest delegation point that can be used for the original QNAME from the cache.

- (1a) For queries with a QTYPE for which the authority only lies at the parent side (like QTYPE=DS), this is the NS RRset with the owner matching the most labels with QNAME stripped by one label. QNAME will be a subdomain of (but not equal to) this NS RRset. Call this ANCESTOR.
- (1b) For queries with other original QTYPEs, this is the NS RRset with the owner matching the most labels with QNAME. QNAME will be equal to or a subdomain of this NS RRset. Call this ANCESTOR.

(2) Initialise CHILD to the same as ANCESTOR.

(3) If CHILD is the same as QNAME, or if CHILD is one label shorter than QNAME and the original QTYPE can only be at the parent side (like QTYPE=DS), resolve the original query as normal, starting from ANCESTOR's name servers. Start over from step 0 if new names need to be resolved as a result of this answer, for example, when the answer contains a CNAME or DNAME [RFC6672] record.

(4) Otherwise, update the value of CHILD by adding the next relevant label or labels from QNAME to the start of CHILD. The number of labels to add is discussed in Section 2.3.

(5) Look for a cache entry for the RRset at CHILD with the original QTYPE. If the cached response code is NXDOMAIN and the resolver has support for [RFC8020], the NXDOMAIN can be used in response to the original query, and stop. If the cached response code is NOERROR (including NODATA), go back to step 3. If the cached response code is NXDOMAIN and the resolver does not support [RFC8020], go back to step 3.

(6) Query for CHILD with the selected QTYPE using one of ANCESTOR's name servers. The response can be:

- (6a) A referral. Cache the NS RRset from the authority section, and go back to step 1.
- (6b) A DNAME response. Proceed as if a DNAME is received for the original query. Start over from step 0 to resolve the new name based on the DNAME target.
- (6c) All other NOERROR answers (including NODATA). Cache this answer. Regardless of the answered RRset type, including CNAMEs, continue with the algorithm from step 3 by building the original QNAME.
- (6d) An NXDOMAIN response. If the resolver supports [RFC8020], return an NXDOMAIN response to the original query, and stop. If the resolver does not support [RFC8020], go to step 3.
- (6e) A timeout or response with another RCODE. The implementation may choose to retry step 6 with a different ANCESTOR name server.

4. QNAME Minimisation Examples

As a first example, assume that a resolver receives a request to resolve foo.bar.baz.example. Assume that the resolver already knows that ns1.nic.example is authoritative for .example and that the resolver does not know a more specific authoritative name server. It will send the query with QNAME=baz.example and the QTYPE selected to hide the original QTYPE to ns1.nic.example.

QTYPE	QNAME	TARGET	NOTE
MX	a.b.example.org	root name server	
MX	a.b.example.org	org name server	
MX	a.b.example.org	example.org name server	

Table 1: Cold Cache, Traditional Resolution Algorithm without QNAME Minimisation, Request for MX Record of a.b.example.org

The following are more detailed examples of requests for an MX record of a.b.example.org with QNAME minimisation, using A QTYPE to hide the original QTYPE and using other names and authoritative servers:

QTYPE	QNAME	TARGET	NOTE
A	org	root name server	
A	example.org	org name server	
A	b.example.org	example.org name server	
A	a.b.example.org	example.org name server	"a" may be delegated
MX	a.b.example.org	example.org name server	

Table 2: Cold Cache with QNAME Minimisation

Note that, in the above example, one query would have been saved if the incoming QTYPE was the same as the QTYPE selected by the resolver to hide the original QTYPE. Only one query for a.b.example.org would have been needed if the original QTYPE would have been A. Using the most-used QTYPE to hide the original QTYPE therefore slightly reduces the number of outgoing queries compared to using any other QTYPE to hide the original QTYPE.

QTYPE	QNAME	TARGET	NOTE
A	example.org	org name server	
A	b.example.org	example.org name server	
A	a.b.example.org	example.org name server	"a" may be delegated
MX	a.b.example.org	example.org name server	

Table 3: Warm Cache with QNAME Minimisation

5. Performance Considerations

The main goal of QNAME minimisation is to improve privacy by sending less data. However, it may have other advantages. For instance, if a resolver sends a root name server queries for A.example followed by B.example followed by C.example, the result will be three NXDOMAINs, since .example does not exist in the root zone. When using QNAME minimisation, the resolver would send only one question (for .example itself) to which they could answer NXDOMAIN. The resolver can cache this answer and use it to prove that nothing below .example exists [RFC8020]. A resolver now knows a priori that neither B.example nor C.example exist. Thus, in this common case, the total number of upstream queries under QNAME minimisation could be counterintuitively less than the number of queries under the traditional iteration (as described in the DNS standard).

QNAME minimisation can increase the number of queries based on the incoming QNAME. This is described in Section 2.3. As described in [devries-qnamemin], QNAME minimisation both increases the number of DNS lookups by up to 26% and leads to up to 5% more failed lookups. Filling the cache in a production resolver will soften that overhead.

6. Security Considerations

QNAME minimisation's benefits are clear in the case where you want to decrease exposure of the queried name to the authoritative name server. But minimising the amount of data sent also, in part, addresses the case of a wire sniffer as well as the case of privacy invasion by the authoritative name servers. Encryption is of course a better defense against wire sniffers, but, unlike QNAME minimisation, it changes the protocol and cannot be deployed unilaterally. Also, the effect of QNAME minimisation on wire sniffers depends on whether the sniffer is on the DNS path.

QNAME minimisation offers no protection against the recursive resolver, which still sees the full request coming from the stub resolver.

A resolver using QNAME minimisation can possibly be used to cause a query storm to be sent to servers when resolving queries containing a QNAME with a large number of labels, as described in Section 2.3. That section proposes methods to significantly dampen the effects of such attacks.

7. References

7.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

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

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

7.2. Informative References

[apnic-qnamemin] Huston, G. and J. Damas, "Measuring Query Name Minimization", September 2020, <https://indico.dns-oarc.net/event/34/contributions/787/attachments/777/1326/2020-09-28-oarc33-qname-minimisation.pdf>.

[devries-qnamemin] de Vries, W., Scheitle, Q., Müller, M., Toorop, W., Dolmans, R., and R. van Rijswijk-Deij, "A First Look at QNAME Minimization in the Domain Name System", March 2019, <https://nlnetlabs.nl/downloads/publications/devries2019.pdf>.

[dnsthought-qnamemin] "Qname Minimisation", October 2021, <https://dnsthought.nlnetlabs.nl/#qnamemin>.

[Huque-QNAME-Min] Huque, S., "Query name minimization and authoritative server behavior", May 2015, <https://indico.dns-oarc.net/event/21/contribution/9>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, DOI 10.17487/RFC6672, June 2012, <https://www.rfc-editor.org/info/rfc6672>.

[RFC6895] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", BCP 42, RFC 6895, DOI 10.17487/RFC6895, April 2013, <https://www.rfc-editor.org/info/rfc6895>.

[RFC7816] Bortzmeyer, S., "DNS Query Name Minimisation to Improve Privacy", RFC 7816, DOI 10.17487/RFC7816, March 2016, <https://www.rfc-editor.org/info/rfc7816>.

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

[RFC8198] Fujiwara, K., Kato, A., and W. Kumari, "Aggressive Use of DNSSEC-Validated Cache", RFC 8198, DOI 10.17487/RFC8198, July 2017, <https://www.rfc-editor.org/info/rfc8198>.

[RFC9076] Wicinski, T., Ed., "DNS Privacy Considerations", RFC 9076, DOI 10.17487/RFC9076, July 2021, <https://www.rfc-editor.org/info/rfc9076>.

[verisign-qnamemin] Thomas, M., "Maximizing Qname Minimization: A New Chapter in DNS Protocol Evolution", September 2020, <https://blog.verisign.com/security/maximizing-qname-minimization-a-new-chapter-in-dns-protocol-evolution/>.

Acknowledgments

The acknowledgments from RFC 7816 apply here. In addition, many participants from the DNSOP Working Group helped with proposals for simplification, clarification, and general editorial help.

Authors' Addresses

Stephane Bortzmeyer
AFNIC
1, rue Stephenson
78180 Montigny-le-Bretonneux
France
Phone: +33 1 39 30 83 46
Email: bortzmeyer+ietf@nic.fr
URI: https://www.afnic.fr/

Ralph Dolmans
NLnet Labs
Email: ralph@nlnetlabs.nl

Paul Hoffman
ICANN
Email: paul.hoffman@icann.org