A Second Look at DNS QNAME Minimization

Jonathan Magnusson[1(✉)], Moritz Müller[2], Anna Brunstrom[1], and Tobias Pulls[1]

[1] Karlstad University, Karlstad, Sweden
{jonathan.magnusson,anna.brunstrom,tobias.pulls}@kau.se
[2] SIDN Labs, Arnhem, The Netherlands
moritz.muller@sidn.nl

Abstract. The Domain Name System (DNS) is a critical Internet infrastructure that translates human-readable domain names to IP addresses. It was originally designed over 35 years ago and multiple enhancements have since then been made, in particular to make DNS lookups more secure and privacy preserving. *Query name minimization* (qmin) was initially introduced in 2016 to limit the exposure of queries sent across DNS and thereby enhance privacy. In this paper, we take a look at the adoption of qmin, building upon and extending measurements made by De Vries *et al.* in 2018. We analyze qmin adoption on the Internet using active measurements both on resolvers used by RIPE Atlas probes and on open resolvers. Aside from adding more vantage points when measuring qmin adoption on open resolvers, we also increase the number of repetitions, which reveals *conflicting resolvers* – resolvers that support qmin for some queries but not for others. For the passive measurements at root and Top-Level Domain (TLD) name servers, we extend the analysis over a longer period of time, introduce additional sources, and filter out non-valid queries. Furthermore, our controlled experiments measure performance and result quality of newer versions of the qmin-enabled open source resolvers used in the previous study, with the addition of PowerDNS. Our results, using extended methods from previous work, show that the adoption of qmin has significantly increased since 2018. New controlled experiments also show a trend of higher number of packets used by resolvers and lower error rates in the DNS queries. Since qmin is a balance between performance and privacy, we further discuss the depth limit of minimizing labels and propose the use of a public suffix list for setting this limit.

Keywords: DNS · Privacy · QNAME minimization · Measurements

1 Introduction

The *Domain Name System* (DNS) is a global hierarchical key/value-store that maps domain names to *Resource Records* (RRs) on the Internet [13,14]. One of the most common use-cases for DNS is performing lookups for RRs containing IP addresses. While privacy and security on the Internet has been a common area of research for the past decades, these topics were less prevalent when DNS was implemented over 35 years ago. The early Internet was relatively small, where everyone knew each other, and the focus was on getting data from point A to point B. Similar to other areas of the Internet, this has resulted in multiple proposals to improve DNS privacy and security as an afterthought. By encrypting DNS traffic with TLS, HTTPS, or QUIC [9–11], it is possible to achieve transport confidentiality. By signing sets of RRs and building chains of trust using DNSSEC [1], it is possible to prove integrity of the data. Due to the hierarchical structure of DNS, the top two levels of servers—root and Top-Level Domain (TLD)—are observing a large portion of non-cached requests on the Internet. From a privacy perspective, it is important that the information sent here is the minimum needed for each task. This is known as the fundamental privacy principle of *data minimization* [6]. The DNS resolvers may strip unnecessary labels for each query in the lookup process. This privacy feature is referred to as *query name minimization* (`qmin`) and is at present standardized in RFC 9156 [4].

The aim of this study is to measure the adoption of `qmin`. To do so, we build upon the experiments of De Vries *et al.* [23], who took a first look at `qmin` adoption in 2018. We considerably extend the experiments, consider an additional source of passive measurements, and include an additional open-source resolver for the controlled experiments. Our contributions are as follows:

1. Extended active measurements from 2018 up until October 2022 show that the adoption of `qmin` has increased from 2.5k resolvers used by RIPE Atlas probes in 2018 to 14k. We also show that the adoption of `qmin` has increased using active measurements on open resolvers, from 18k open resolvers categorized as `qmin`-enabled in 2018 to 80k in 2022.
2. Extended passive measurements at root and TLD name servers show an increase of `qmin` adoption from 0.6% in 2018 to 2.5% at one root server and from 35.5% in 2019 to 57.3% at the `.nl` TLD. We also observe that a significant amount of noise is removed when filtering out invalid labels at root.
3. Up-to-date performance and error-rate measurements of four open source resolvers (Bind, Knot, PowerDNS, and Unbound) show that while error-rates have been significantly reduced for all resolvers, the number of packets have gone both up (Bind and Unbound) and down (Knot).
4. Grounded in our findings, we discuss the trade-off between privacy and performance of minimizing queries, and propose that a promising solution may be to set the depth limitation of `qmin` using a public suffix list.

The rest of the paper is structured as follows. Section 2 provides background with regards to DNS and `qmin` as well as related work on `qmin`. We present active measurements surveying `qmin` adoption from the client-side perspective in Sect. 3. Section 4 presents passive measurements surveying `qmin` adoption from the name server perspective. Controlled experiments measuring the performance and error rates of resolvers with `qmin` implemented are shown in Sect. 5. Section 6 discusses our findings, focusing on the observed resolver behavior and the depth limit of minimizing queries. Finally, Sect. 7 concludes the paper.

2 Background and Related Work

2.1 The Domain Name System

In order for computers to communicate on the Internet, they need to indicate where a packet of data should be sent. This is solved by assigning addresses to devices within the Internet Protocol (IP) using IP addresses [18]. In order for end users to navigate on the Internet, names are mapped to IP addresses. For ARPANET, the Internet's predecessor, this mapping was originally maintained in a file on each computer. With the explosive growth of machines on the network, it quickly became unfeasible to maintain such a file. In 1985 a hierarchical system of names was introduced called the *Domain Name System* (DNS) [13,14]. DNS is responsible for mapping a domain name to a *Resource Record* (RR). The most common use of DNS is to request the associated `A` or `AAAA` RR of a domain, corresponding to the IPv4 and IPv6 addresses, respectively. Other common RRs include `MX` for identifying mail servers for a domain, `TXT` for associating arbitrary text with a domain, and `NS` for referring a resolver to the authoritative name servers for the requested zone. The distributed hierarchy is divided into "zones" to allow for local maintenance. The zones are connected in a tree hierarchy and each zone is served by multiple authoritative name servers: *root* name servers, *Top-Level Domain* (TLD) name servers, etc.

When querying for the IPv4 address of a domain name, a DNS client stub resolver sends a query to a recursive DNS resolver, requesting the `A` RR for the specified domain. Using the client-server model, the DNS resolver sends a query to one of thirteen[1] root name servers. Instead of responding with the `A` record containing the requested IP address, the root name server sends a referral with an `NS` record containing the IP address of the matching TLD name server. TLDs include generic TLDs (e.g., `.com`, `.net`, `.org`), country-code TLDs (e.g., `.se`, `.nl`, `.us`) and sponsored TLDs (e.g., `.edu`, `.gov`, `.mil`). The DNS resolver then sends the query to the specified TLD name server. Yet again, the DNS resolver will not get the `A` record requested, but instead a referral with the `NS` record to another name server. Depending on the number of delegations, this will continue until the correct RR (in this case an `A` record IP address) is obtained from the name server that is authoritative for that record. When the recursive DNS resolver has obtained the RR, it will forward the response to the client stub resolver that initiated the process. Resolvers are also able to cache responses for a time in order to improve performance and reduce the amount of DNS traffic.

2.2 Query Name Minimization

There is no added functional value when sending a query containing all labels to the root. This was the fundamental idea of RFC 7816 *query name minimization* (`qmin`) specified in March 2016 [3]. A `qmin`-enabled DNS resolver querying the domain name `www.example.domain` will only send a query for the right-most label (`.domain`) to the root name server. This minimizes the amount of information being leaked. When the resolver receives the `NS` RRs

[1] https://www.iana.org/domains/root/servers.

of the TLD name servers for `.domain` it appends one additional label to the query (`example.domain`) and sends it to the TLD name server. The resolver will then receive the address to the authoritative name server for `example.domain` and finally append the third and last piece of the domain name (`www.example.domain`), send the last request and, hopefully, get the requested RR in return.

The default implementation of `qmin` in RFC 7816 has two main challenges. Firstly, the standardized RR type for queries when using `qmin` is the `NS` RR, which could cause some name servers to not respond or result in an error if no such RR is found for a minimized query. This behavior is not according to the standard, but an error on the name server side. Secondly, a domain name with many labels creates additional traffic, which could be abused for Denial-of-Service (DoS) attacks. If the DNS zone `example.domain` contains a RR for `a.b.c.d.example.domain`, a `qmin`-enabled resolver would send multiple queries to `example.domain` name servers before asking for the final RR. This has led to alternative implementations of `qmin` in the wild, which includes requesting `A` RRs instead of `NS` and iteratively adding *multiple* labels after the second-level label instead of one at a time.

In November 2021, RFC 7816 was obsoleted by RFC 9156 [4], which combined the results and recommendations from De Vries *et al.* with experiences from implementing `qmin` in the wild. Updated implementation details were presented to reduce error rates and improve performance while keeping a reasonable level of privacy. The `NS` RR was discarded in favor of `A` and `AAAA` RRs when sending minimized queries. Fallbacks for specific error codes were specified and two tunable parameters for incrementally adding labels were introduced. The RR types used for queries in standard DNS, RFC 7816 and RFC 9156, respectively, are shown in Table 1.

Table 1. DNS queries and responses of Standard DNS, RFC 7816 and RFC 9156.

Standard DNS	`qmin` RFC 7816	`qmin` RFC 9156
a.b.foo.bar. `A` $\rightarrow$.	bar. `NS` $\rightarrow$.	bar. `A` $\rightarrow$.
bar. `NS` $\leftarrow$.	*bar.* `NS` $\leftarrow$.	*bar.* `NS` $\leftarrow$.
a.b.foo.bar `A` $\rightarrow$ bar.	foo.bar `NS` $\rightarrow$ bar.	foo.bar `A` $\rightarrow$ bar.
foo.bar `NS` $\leftarrow$ *bar.*	*foo.bar* `NS` $\leftarrow$ *bar.*	*foo.bar* `NS` $\leftarrow$ *bar.*
a.b.foo.bar `A` $\rightarrow$ foo.bar.	b.foo.bar `NS` $\rightarrow$ foo.bar.	b.foo.bar `A` $\rightarrow$ foo.bar.
a.b.foo.bar `A` $\leftarrow$ *foo.bar.*	*b.foo.bar* `NS` $\leftarrow$ *foo.bar.*	*b.foo.bar* `NS` $\leftarrow$ *foo.bar.*
	a.b.foo.bar `NS` $\rightarrow$ foo.bar.	a.b.foo.bar `A` $\rightarrow$ foo.bar.
	a.b.foo.bar `NS` $\leftarrow$ *foo.bar.*	*a.b.foo.bar* `A` $\leftarrow$ *foo.bar.*
	a.b.foo.bar `A` $\rightarrow$ foo.bar.	
	a.b.foo.bar `A` $\leftarrow$ *foo.bar.*	

There are two modes called "relaxed" and "strict" when enabling `qmin` on a resolver [2,17]. In "relaxed mode", the resolver will fall back to querying for the full query name to potentially broken name servers. In contrast, the "strict mode" will not, and it therefore results in more non-resolved domains.

2.3 Related Work

A first measurement study of the adoption of qmin was done by De Vries *et al.* [23], where they measured from April 2017 to October 2018. By carrying out both active and passive measurements they could conclude that there was a slow but steady adoption of qmin, which resulted in noticeable improvements of query privacy at root and TLD name servers. Between two active measurements they discovered an improved method to measure qmin adoption. A qmin-enabled resolver might forward the query to a not qmin-enabled resolver. If a minimized query is cached at the latter, it could incorrectly be classified as a qmin-enabled resolver. A wildcard label was therefore introduced to make each query in the subsequent active measurements unique. Fingerprinting the resolver algorithms showed that the implementation of qmin in the wild differed from the details in RFC 7816 [3] in order to reduce error rates and improve performance. Further, controlled experiments with three open source resolvers measured the error rates and performance of using qmin in different modes. From the results in the study, new implementation recommendations were designed by combining the best practices observed in the wild.

ICANN (Internet Corporation for Assigned Names and Numbers) has been doing measurements on root traffic to analyze leaks, patterns and characteristics of resolvers [12]. They classify a query at the root as minimized if it consists of only one single label and clarify that non-valid queries (TLDs) are filtered out. A resolver is only classified as qmin-enabled if *all* requests originating from it are minimized.[2] The passive measurements of root traffic by ICANN is used to categorize resolvers by counting the number of labels for each query. The passive measurements of root traffic in Sect. 4.2 in this study is not for classifying resolvers, but instead for observing the total share of minimized queries over time.

A report from CZ.NIC, the domain registry for the .cz ccTLD, measured and analyzed qmin support in the .cz DNS ecosystem [7]. They introduce a new method of classifying qmin-enabled resolvers with machine learning. While the new method is promising, they noted a couple of circumstances where the prediction could be distorted. There could be multiple resolvers behind one single IP address, and sometimes there are too few queries from a single resolver to make a good enough prediction. A two label query at the TLD name server does also not necessary mean that the query is minimized. This limitation is true for this study as well. A resolver could be falsely classified as a qmin-enabled resolver when the fully-qualified domain name only has two labels. The passive measurements of TLD traffic by CZ.NIC is used to categorize resolvers using machine learning. The passive measurements of TLD traffic in this study is for validating queries from already classified resolvers (Sect. 4.1) and for observing the total share of minimized queries over time (Sect. 4.2).

[2] https://ithi.research.icann.org/about-m3.html#M3Res.

3 Active Measurements

The goal of the active measurements is to query resolvers on the Internet in order to observe the adoption of `qmin` based on their responses. The active measurements consist of two parts: resolver adoption over time (Sect. 3.1) and adoption by open resolvers (Sect. 3.2). The former classifies the resolvers used by RIPE Atlas probes [21] and the latter classifies resolvers from a list generated by scanning the IPv4 address space for servers listening on UDP port 53 [20]. The purpose of the first active measurement is to see the adoption trend of `qmin` over time and to observe characteristics of the resolvers adopting `qmin`. The purpose of the second active measurement is to classify open resolvers and then use these results in the passive measurements (Sect. 4.1) to enhance the classification accuracy. We also observe the adoption of `qmin` on open resolvers since the previous `qmin` adoption study by De Vries *et al.*.

3.1 Resolver Adoption over Time

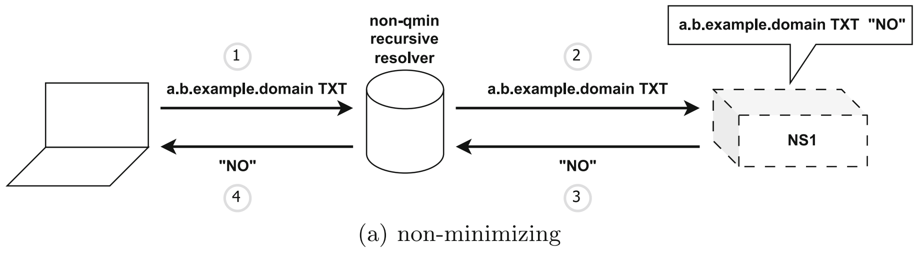
Method. In this study, the Internet-wide active measurement technique introduced by De Vries *et al.* to identify support for the `qmin` feature in recursive DNS resolvers was employed. The method leverages the fact that resolvers without `qmin` will transmit the complete request to the authoritative name server, while resolvers with `qmin`-enabled will iteratively add labels to the request.

To detect `qmin` support, two authoritative name servers, both running mainstream server implementations, were specially configured to behave in the following way, as illustrated in Fig. 1. When a non-minimizing resolver is queried for the TXT record at `a.b.example.domain` (Fig. 1(a)), it sends a query with the full query name and type TXT to server ns1. In this case, ns1 responds with an authoritative answer containing a TXT RR with the text “NO”; ns2 is never queried. However, when a minimizing resolver is queried for the same name (Fig. 1(b)), it sends a query with query name `b.example.domain` and type NS or A to server ns1. In this case, ns1 responds with a referral to the server ns2. Server ns2 responds with a TXT RR containing “HOORAY”. This setup makes it possible to detect the `qmin` functionality of a resolver from the client-side (see Fig. 2).

Since the first look at `qmin` by De Vries *et al.*, active measurements using RIPE Atlas probes [21] have been continued by NLnet Labs[3] and the measurement results have been presented as part of DNSThought [16].

In Sect. 3.1 we do not conduct any active measurements of our own, but analyze and discuss already available data. Specifically, we utilize `qmin` measurement data collected by NLnet Labs using RIPE Atlas and made available for analysis on the DNSThought website. The graphs in this study that are generated from this data are annotated thus: “(data source: DNSThought)”.

[3] https://nlnetlabs.nl/.



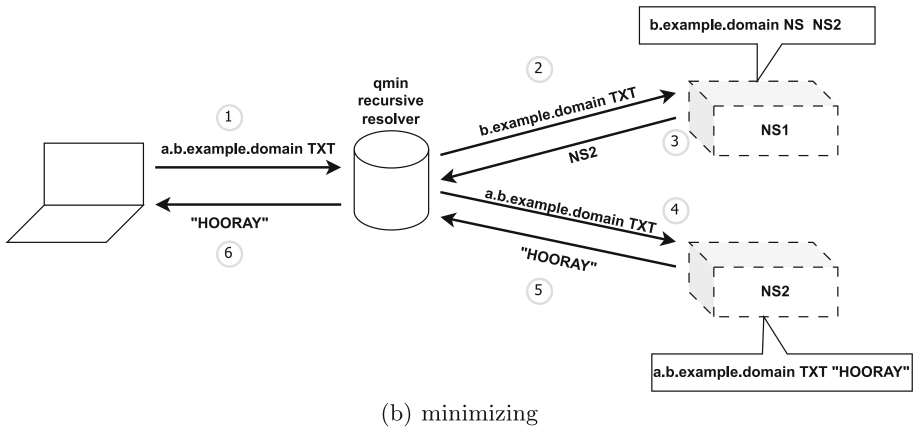


Fig. 1. Example diagram of detecting a (non-)minimizing resolver from the client-side using two authoritative name servers.

Results. Figure 3 shows the current trend of `qmin` adoption from the client-side (RIPE Atlas probes) perspective. Green represents the number of `qmin`-enabled resolvers and orange the number of not `qmin`-enabled resolvers. The gray in turn shows the resolvers which are not answering to the `qmin` measurements, but still responds to other queries done as part of DNSThought. The RIPE Atlas probes are churned daily in batches and a bug in the locking system of the measurement caused newly added probes to not query for `qmin`. This caused a steady increase of gray resolvers from early 2020 to early 2022. With the help of NLnet Labs we contacted RIPE NCC and the bug was fixed on the 6th of April 2022.

In order to see the relative adoption of `qmin`-enabled resolvers we created Fig. 4, based on the assumption that the out-churned probes are not correlated with the `qmin` adoption of their resolvers. We see that 64% of the resolvers used by RIPE Atlas probes in 2022 have enabled `qmin` compared to 10% around the time of the report of De Vries *et al.* at the end of 2018. When going back to Fig. 3, we still see the slight increase of `qmin`-enabled resolvers in April 2018 that was pointed out by De Vries *et al.* in the original study and attributed to Cloudflare enabling `qmin` on their DNS resolvers by default. There has been a steady increase of `qmin`-enabled resolvers since then, until a spike in January 2020 after which the adoption was seemingly slowing down. But looking at Fig. 4,

```
$ dig @1.1.1.1 a.b.qnamemintest.net TXT +short
"HOORAY - QNAME minimisation is enabled on your resolver :)!"

$ dig @8.8.8.8 a.b.qnamemintest.net TXT +short
"NO - QNAME minimisation is NOT enabled on your resolver :("
```

Fig. 2. Using dig to query resolvers for qmin.

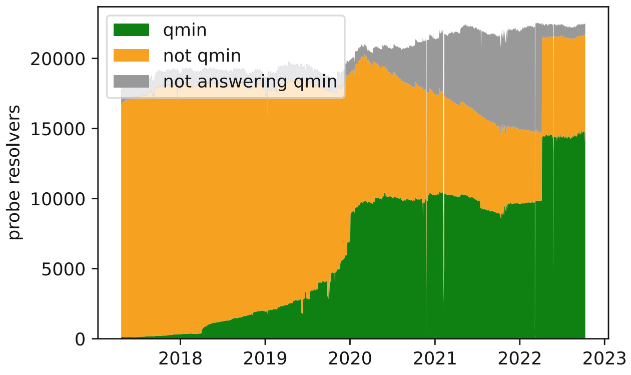


Fig. 3. Adoption of qmin over time (data source: DNSThought [16]).

we know that the relative adoption of qmin-enabled resolvers is actually still going up. While only a small fraction of resolvers were using qmin at the time of the study by De Vries *et al.*, our results suggest that a majority of resolvers are today qmin enabled.

The sudden spike of adoption in 2020 prompted a deeper dive into the characteristics of the qmin-enabled resolvers observed by DNSThought. Looking at the top ten ASNs of qmin-enabled resolvers (see Fig. 5), we observe that Google has become the biggest ASN of qmin-enabled resolvers since January 2020.

The RIPE Atlas probes used in DNSThought have been using the zone a.b.qnamemin-test.internet.nl for measuring qmin adoption. This means that the improved method of using a wildcard label in order to mitigate cached delegations (see Sect. 2.3), is *not* utilized. Some resolvers may therefore have a cached minimized query from a previously forwarded request and show up as qmin-enabled at DNSThought. By contacting NLnet Labs, a new zone was set up for measuring the adoption by open resolvers in Sect. 3.2 using this improved method.

3.2 Adoption by Open Resolvers

Method. In the initial study by De Vries *et al.*, a list of 8 million addresses from Rapid7 was used [20] to query open resolvers for qmin adoption. Rapid7 generated this list by scanning the Internet for servers responding on UDP port

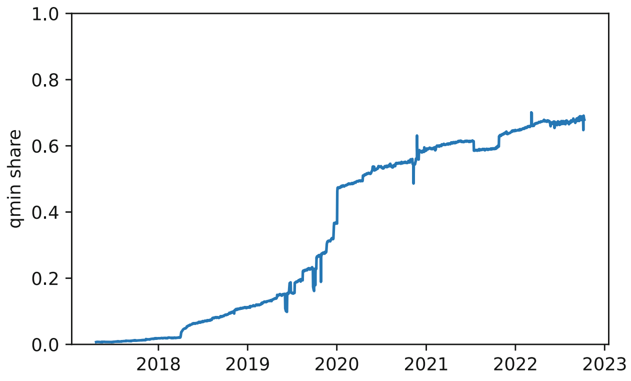


Fig. 4. Share of `qmin` over time (data source: DNSThought [16]).

53. For the active measurements, requests for a `TXT` RR were sent to a zone under NLnet Labs' control through each of the resolvers on the list to classify them as either `qmin`-enabled or not. The flowchart in Fig. 6 shows the process of the classification. First a query is sent to the resolver, which will either respond or timeout. If it does not timeout the answer is checked for errors. If the response is free from errors the next check is for a correct answer. A correct answer is a `TXT` record that contains either "HOORAY" or "NO". Finally the response is classified as either of those two. The results from the previous study are included in Sect. 3.2 to allow for comparison.

For this study we used the Rapid7 list from February 2022 since access to the list was later restricted.[4] The list contains 6 million addresses and was used in April 2022 to send queries from North Virginia, Tokyo, and Frankfurt using EC2 instances on Amazon Web Services to see if the geographical location had any effect on the results. We sent 100 queries to each resolver to collect more data points and get a more comprehensive view of each resolver. However, the system described previously has the following limitation. The delegation (i.e., the NS records from the referral response) from a test might be cached by a resolver that performs `qmin`, such that the outcome of a subsequent test to the same resolver favors `qmin`. To overcome this limitation, we developed a custom authoritative server that behaves similarly to the other system, but additionally allows custom query names, the referrals for which are synthesized, based on the query. This allows us to send unique query names in close proximity by using a wildcard label, avoiding the effects of cached delegations. For example, a query for `a.tokyo-00.qnamemintest.net` (corresponding to the first iteration of queries from Tokyo) results in a query of `tokyo-00.qnamemintest.net` to ns1 by a qmin resolver. In response, ns1 is able to refer the resolver to ns2 for `tokyo-00.qnamemintest.net`. When the next iteration of queries from Tokyo is sent, `tokyo-01.qnamemintest.net` will not be found in the cache.

[4] https://web.archive.org/web/20220414114758/https://www.rapid7.com/blog/post/2022/02/10/evolving-how-we-share-rapid7-research-data-2/.

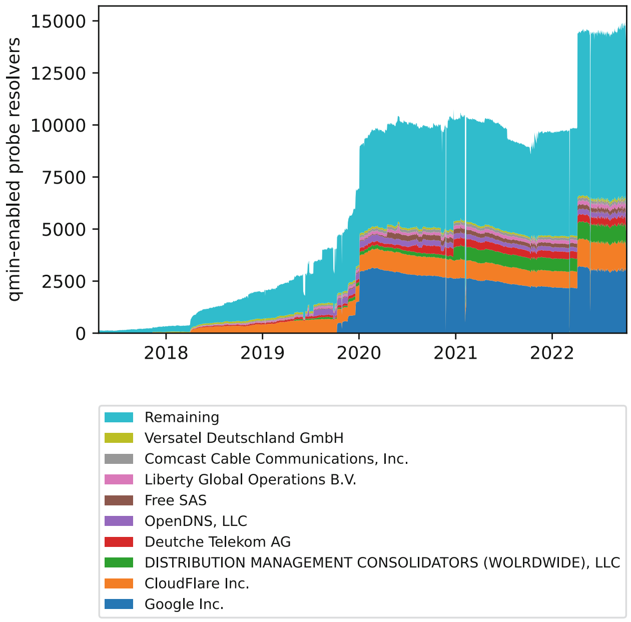


Fig. 5. Top ten `qmin`-enabled resolver ASNs (data source: DNSThought [16]).

Table 2. Categorized responses from open resolvers.

Year	Geo	#resolv	#queries	Resp	Noerr	Correct	`qmin`
2018	Netherlands	8M	8M	64%	32%	72%	1.6%
2022	Nvirginia	6M	600M	70.82%	19.46%	78.12%	16.43%
2022	Tokyo	6M	600M	70.71%	19.45%	78.32%	16.42%
2022	Frankfurt	6M	600M	70.78%	19.45%	78.21%	16.42%

Results: Over Time and Location. Table 2 shows the results for our measurements (2022) from North Virginia, Tokyo, and Frankfurt as well as the earlier results from 2018 by De Vries *et al.*. Our results are calculated from all 100 queries to each of the 6 million resolvers. The values therefore represent the fraction of queries and *not* the fraction of resolvers. The columns `year` and `geo` indicate which study and which geographical location the queries were sent from. The `#resolv` column shows the size of the list from Rapid7 and the `#queries` shows the total number of queries sent. The `resp` column shows the percentage of queries that did not timeout. The column named `noerr` shows the percentage of queries *from responding resolvers* that did not contain any errors (e.g., `SERVFAIL`, `NXDOMAIN` and `REFUSED`). The `correct` column shows the percentage *of noerror responses* that contained a correct `TXT` RR reply, which means that the response was either “HOORAY” or “NO”. The last column, `qmin`, shows the percentage of “HOORAYs” out of the total number of correct responses.

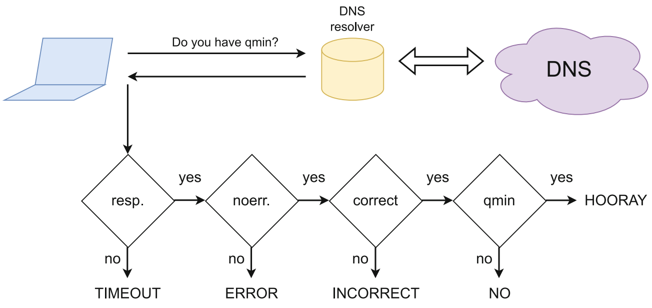


Fig. 6. Flow-chart of categorizing `qmin` query responses from resolvers.

The share of minimized queries in 2018 was 1.6% and in 2022 this number increased to about 16%, measured from three different geographical locations. Comparing the geographical locations we observe minimal deviations in the results. The Rapid7 scan on UDP port 53 in February 2022 resulted in 6 million addresses, which is a decrease of 25% from 2018, and the active measurements of this study using that list shows that the share of non-timeouts has increased to almost 71% from 64%. Another interesting observation is that the share of `NOERROR` replies had gone down from 32% to around 19% and more than 90% of the errors are `REFUSED`. The reason could be that some resolvers are configured to only handle queries from clients within a specified subnet. The share of correct `TXT` responses have increased from 72% to 78% and finally the share of queries classified as minimized have increased from 1.6% to 16%. So while we only got correct responses from a small fraction of the open resolvers, we see a ten-fold increase in the use of `qmin` also in this data set.

Results: Conflicting Resolvers. The original study by De Vries *et al.* sent a single query to each resolver and classified each resolver based on the response. In this study we queried each address on the Rapid7 list 100 times, which revealed additional information when classifying the resolvers. First, the resolvers that did not respond with a single correct `TXT` RR were filtered out (5.27M). All responses from these resolvers were a mix of timeouts, errors and incorrect `TXT` RRs. Incorrect `TXT` responses contained: nothing, validation tokens, notifications about the domain being expired or disabled, and replies such as: "txt", "this is a txt record", "hello, dns!", "OK" and "pong". The remaining more interesting resolvers contained at least one correct answer, which could either be a HOORAY or a NO reply. Our analysis of these 730k resolvers found that 87.3% responded correctly to between 80–100 queries, while the remaining 100k resolvers responded correctly to 1–79 queries. Further research is needed to determine the reasons for this discrepancy and its potential impacts on the overall quality of replies from open resolvers.

One interesting observation is that a subset of the resolvers did not consistently respond with only HOORAY or only NO, but responded with at least one of each

Table 3. Comparing share of classified resolvers

Year	#resolv	`qmin` (%)	Not-`qmin` (%)	Conflicting (%)
2018	8M	18.8k (0.2%)	1.1M (13.7%)	N/A
2022	6M	80.2k (1.3%)	539.3k (8.9%)	120.8k (2%)

during the 100 queries. So in addition to the list of *qmin-enabled resolvers* and the list of *not qmin-enabled resolvers*, we consider a list of resolvers which sometimes answered HOORAY and at other times NO. This list is called *conflicting resolvers*. A resolver is classified as `qmin`-enabled if at least one query resulted in a HOORAY and none of the queries resulted in a NO. A resolver is classified as not `qmin`-enabled if at least one query resulted in a NO and none of the queries resulted in a HOORAY. Finally, a resolver is classified as conflicting if at least one query resulted in a HOORAY and at least one query resulted in a NO.

In the original study by De Vries *et al.* 0.2% of 8 million resolvers were classified as `qmin`-enabled (see Table 3). In this study we classified 1.3% of 6 million resolvers as `qmin`-enabled. In the original study 13.7% of resolvers were classified as not `qmin`-enabled, whereas 8.9% of resolvers were classified as not `qmin`-enabled in this study. Additionally 2% were classified as the new category of conflicting in this study.

Table 4. Share of conflicting resolvers, top 10 countries.

Country	CN	RU	US	BR	ID	AU	UA	IR	PL	ZA
Share	26.8%	11.4%	5.5%	4.5%	3.6%	3.4%	3.0%	2.4%	2.3%	2.1%

The minimum number of correct replies in a conflicting resolver is two: one HOORAY and one NO. This results in a 50% share of `qmin`. However, when 100 queries are sent, the ratio of HOORAY and NO responses can range from 1:99 to 99:1. Our analysis showed that out of approximately 109k conflicting resolvers, 80k (73.4%) had a `qmin` share of less than 50%. Approximately 8k resolvers (7.4% of the total) had a `qmin` share of more than 90%. Additionally we looked into the geographical location of ASes containing the conflicting resolvers. Table 4 shows the top 10 countries according to their share of 120.8k conflicting resolvers based on ASN whois lookups. China, Russia and the United States dominate this list.

We also wanted to see if the conflicting resolvers were more or less short-lived compared to the resolvers categorized as `qmin`-enabled and not `qmin`-enabled. Five months after the initial measurements (September 2022) we observed that 33.1% of the `qmin`-enabled resolvers no longer responded. We also got time-outs from 38.2% of the not `qmin`-enabled resolvers and 35.5% of the conflicting resolvers. Thus slightly more than a third of the resolvers were no longer reachable, but we could not observe any major difference in lost resolvers across our categories.

Results: Unexpected Google. Given the rapid adoption of `qmin`-enabled resolvers from the Google ASN since 2020 seen in Fig. 5, it was unexpected to see that the Google Public DNS resolvers were classified as *not* `qmin`-enabled in the active measurements of open resolvers above. We performed additional queries to verify this behavior of the `8.8.8.8` and `8.8.4.4` Google Public DNS resolvers using three different zones: `a.b.qnamemin-test.nlnetlabs.nl`, `a.b.qnamemin-test.internet.nl`, and `a.b.qnamemintest.net`.

The first zone is a set of subdomains under the second-level domain of NLnet Labs that was used for measuring `qmin` adoption in the study by De Vries *et al.*. The second zone is the official name for measuring `qmin` after the publication of the original study.[5] This is also the zone used by DNSThought at NLnet Labs. The third zone was set up in early 2022 for this study, using the label wildcard cache mitigation technique, which neither of the other two zones implemented. All of these zones are using the same method when measuring `qmin` from the client-side. When using Google Public DNS resolvers, only `a.b.qnamemin-test.internet.nl` responded with HOORAY (see Fig. 7), which is the name used for the RIPE Atlas `qmin` adoption measurements in DNSThought.

```
$ dig @8.8.8.8 a.b.qnamemin-test.nlnetlabs.nl TXT +short
"NO - QNAME minimisation is NOT enabled on your resolver :("

$ dig @8.8.8.8 a.b.qnamemin-test.internet.nl TXT +short
"HOORAY - QNAME minimisation is enabled on your resolver :)!"

$ dig @8.8.8.8 a.b.qnamemintest.net TXT +short
"NO - QNAME minimisation is NOT enabled on your resolver :("
```

Fig. 7. Using dig to query a Google Public DNS resolver (8.8.8.8) for `qmin`.

By contacting NLnet Labs we were told that Google had reached out in May 2020 in regards to `qmin`. They wrote that they had implemented `qmin` but with a depth limitation that stops after two labels. This would result in a partial `qmin` that sends minimized labels to the roots and TLDs but not to Second-Level Domains (2LDs) such as `co.uk`. They also said that they would like to extend it to public suffix plus one label in the future. As a result, the Google Public DNS does not show up as minimizing queries on DNSThought at all, which is why they added an exception to the depth limitation for `a.b.qnamemin-test.internet.nl` to reflect that the resolvers do minimize queries (up to a point). They did not want to "cheat" the system, but still get credit for the privacy benefit of minimizing queries at the root and TLD level. This brings to question what an adequate level of minimizing queries is in regards to performance and privacy, which is further discussed in Sect. 6.

[5] https://web.archive.org/web/20220428083123/https://labs.ripe.net/author/wouter_de_vries/making-the-dns-more-private-with-qname-minimisation/.

4 Passive Measurements

In this section we show how `qmin` has evolved in the years between the study by De Vries *et al.* and October 2022 on a larger scale. As the previous study, we rely on data collected at the root servers as well as the `.nl` ccTLD. Furthermore, we improve the measurement technique, dive deeper into `qmin` adoption, showing who drives `qmin` and who lags behind, and find that `qmin`-enabled resolvers can leak information occasionally.

4.1 Method

In order to identify minimized queries and `qmin`-enabled resolvers in our passive data sets, we rely on a similar methodology as De Vries *et al.* but extend and improve it further.

Identifying Minimized Queries. We count queries as minimized if they contain one label at the root and two at `.nl`. De Vries *et al.* already mentioned the fact that the data from the root might be influenced by queries to non-existing, single-label, domain names caused by Google's Chrome browser. Including these queries in our analysis would lead to overestimating the number of minimized labels at the root.[6] For this reason we filter out these queries by considering only queries with one and, respectively, two labels for existing domain names.

The DITL (a Day In The Life of the Internet) data collected by DNS-OARC[7] (Operations Analysis and Research Center) does not contain DNS responses. For this reason, we verify for each query that contains only one label whether the queried label belongs to a registered top level domain. The data sets of `.nl` contain both DNS queries and responses. This allows us to filter for queries that result in a DNS response with response code NOERROR.

Identifying qmin-Enabled Resolvers. In order to single out resolvers that have enabled `qmin`, we again extend the methodology proposed by earlier work. Here, we use traffic from resolvers that we identified as `qmin`-supporting in the previous section as ground truth. The fact that we now also differentiate between conflicting resolvers allows us to identify `qmin`-enabled resolvers in passive data with a smaller error margin. To address possible biases from our Dutch vantage point, we now also take data from the Swedish ccTLD `.se` into account.

Figure 8 and Fig. 9 show the share of minimized queries of resolvers in the different categories to the name servers of the two ccTLDs. Only resolvers with a minimum of ten observed queries at each ccTLD that date were included.

The share of minimized queries for `qmin`-enabled resolvers is over 90% (median) and is in stark contrast with resolvers that have not enabled `qmin`. Those resolvers send less than 20% of their queries minimized. As expected, the conflicting resolvers show more diverse behaviour. For the remainder of

[6] For more details we refer to the bug report [8] and to Verisign [22].
[7] https://www.dns-oarc.net/.

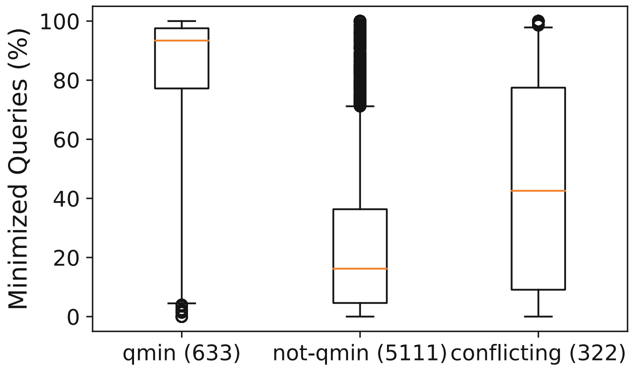


Fig. 8. Minimized queries to the `.nl` ccTLD. Whiskers at 5th and 95th percentiles.

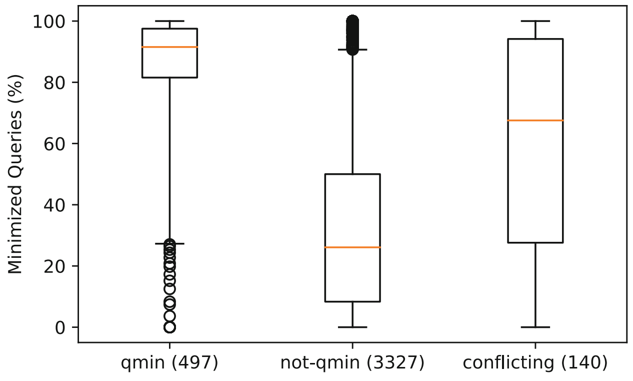


Fig. 9. Minimized queries to the `.se` ccTLD. Whiskers at 5th and 95th percentiles.

this section, we classify resolvers as `qmin`-enabled if they send at least 77.2% of their queries minimized to name servers of 2nd level domain names (25% quartile in Fig. 8). The chosen threshold reduces the number of wrongly classified resolvers, since it is above the 95th percentile of minimized queries of non-`qmin` enabling resolvers and above the 75th percentile of minimized queries of conflicting resolvers in the `.nl` data set.

4.2 Results

Since 2019, `qmin` adoption has improved significantly, at least from the perspective of TLDs. Figure 10 shows the share of queries sent to the K-Root servers and to three of the four `.nl` authoritative name servers. The blue color indicates the share of queries regardless of whether the domain name exists or not. The yellow color marks measurements that include only minimized queries to existing domain names.

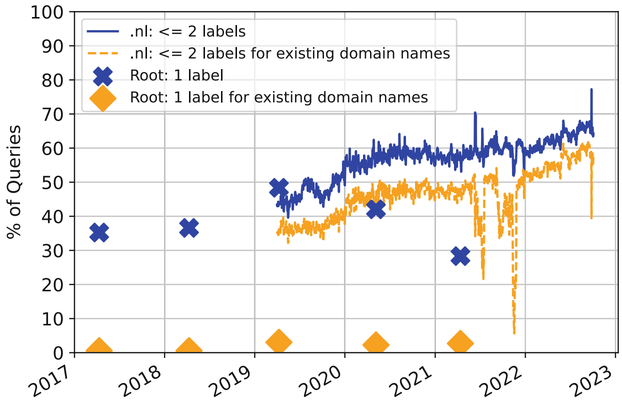


Fig. 10. Minimized queries to the `.nl` ccTLD and K-Root over time.

The increase in minimized queries at `.nl` is clearly visible and has now reached 64% (compared to 43% in 2019). Adoption raises regardless of whether or not we filter out queries for non-existing domain names. Occasional drops in minimized queries are caused by random events, for example crawlers or misconfigurations. The picture at the root is, however, less clear.

Results: Impact of Non-existing Domain Names. If we look at queries for all domain names at the root, then adoption of `qmin` has even decreased in the last years. In 2017, 35.2% of queries would have been minimized compared to 28.3% in 2021. This shows that filtering out queries to non-existing domain names is important when measuring `qmin` at the root. When doing so the share of minimized queries at the root decreases drastically. In 2021, the share drops from 28.3% to 2.7%.

In 2021, we can see clearly what causes the high number of potentially minimized queries. Then, the number of queries with one label decreased from 42% to 28%. This drop correlates with the rollout of a new Chromium version for Android in November 2020. From then on, the feature responsible for sending out the random subdomain queries has been disabled. Root server operators noticed this rollout [22]. This shows that the numbers reported by De Vries *et al.* were indeed heavily influenced by Chrome's behaviour.

Omitting queries to non-existing domain names paints a clearer picture of the deployment of `qmin`. When doing so the share of minimized queries at the root increases from 0.4% in 2017 to 2.7% in 2021.

Results: Qmin Adoption in Detail. Not everyone benefits equally from the rising `qmin` adoption. Already 2021 the root traffic exhibits strong local variations. The K-Root site in Bhutan receives 17.8% of its queries minimized, while at the site in Tajikistan only 0.1% of the queries fall into this category.

We dive deeper into this phenomenon relying on data collected at the `.nl` name servers on October 4 2022.[8] We map each IP address to its corresponding country and autonomous system (AS) using the Maxmind database. On this date, we observe 2.9B queries from 1.4M unique IP addresses from 236 countries and 38,469 ASes. We count an IP address as a `qmin`-enabled resolver if the share of existing queries is above the threshold defined above. This reflects the top 75% resolvers that have enabled `qmin` in our ground truth data set (see Fig. 8).

By Country. Only countries from which queries have been received from at least 100 unique IP addresses are taken into account. This leaves us with IPs from 159 countries. Interestingly, the country with the highest share of minimized queries is Yemen (see Table 5). This high share of minimized queries is mainly driven by large providers. Here, 10% of resolvers have `qmin` enabled, but those are responsible for a large share of queries from this country. We explore the influence of single networks in the next section. Overall, however, these countries only account for a small fraction of total traffic at `.nl`.

Table 5. Countries with most minimized `.nl` queries.

Country	`.nl` query share	Minimized queries	Qmin enabled resolvers
Yemen	0.01%	89.2%	10.0%
Afghanistan	0.01%	85.7%	19.5%
Iraq	0.01%	84.5%	45.3%
Benin	0.01%	82.2%	22.0%
Finland	1.0%	81.9%	33.6%

The adoption rate is lower when we look at the countries from which `.nl` receives the most queries. Table 6 summarizes the results. For these countries, the share of minimized queries vary between 1% for China and 59.5% for Great Britain. Also the share of `qmin`-enabled resolvers is the highest in Great Britain, followed by the Netherlands.

Table 6. Deployment of `qmin` from the top 5 origins of `.nl` queries.

Country	`.nl` query share	Minimized queries	Qmin enabled resolvers
US	35.3%	32.3%	19.2%
Netherlands	20.6%	41.6%	28.9%
Germany	9.2%	14.4%	25.8%
China	4.6%	1.0%	4.4%
Great Britain	3.2%	59.5%	30.9%

By Network. Within one country there are significant differences. As an example, we have a closer look at networks with IP addresses located in the Netherlands,

[8] We also carried out our analysis two months earlier, with similar results.

and focus especially on networks of Internet Service Providers (ISPs). We rely on a community maintained list of ISPs[9] to identify relevant networks. ISPs serve especially many human users, who would potentially benefit the most from `qmin`.

In our dataset, 23 networks belong to an ISP and send queries from at least 10 distinct addresses. Of those, less than half (10 networks, 43.5%) send their queries through `qmin` enabled resolvers most of the time. Networks that use `qmin` include Freedom Internet (AS 206238), an ISP that positions itself as an especially secure and "free" ISP. Also T-mobile Netherlands (AS 13127), one of the largest ISPs in the Netherlands, appears to have `qmin` enabled. The former incumbent KPN (ASes 1136, 8737, and 15879) does not appear to have enabled `qmin` on their resolvers (minimizing resolvers send only 0.3% of the queries).

Qmin in the Context of Other Internet Standards. Qmin has became best common practice. Overall, resolvers that have enabled `qmin` show more support for "modern" Internet standards and DNS best practices. We compare resolvers that send queries to the `.nl` name servers via IPv6, that indicate support for DNSSEC[10], and that indicate a EDNS(0) buffer size of 1,232 bytes[11] with resolvers that do not follow these best practices.

Table 7 shows that resolvers that have enabled IPv6, show support for DNSSEC, and set the recommended buffer sizes also support `qmin` in most cases. These resolvers reflect 3.7% of all resolvers observed in our dataset. In contrast, only a minority of resolvers that do not follow any of these best practices have enabled `qmin` (4.8% of all resolvers in our dataset). The largest group of resolvers do indicate support for DNSSEC, but rely on legacy IPv4, and signal a different buffer size. Of those, 16.4% send minimized queries (63% of all resolvers in our dataset).

Table 7. Qmin support per resolver by support of modern standards and best common practices.

Standard supported	IPv6	DNSSEC	Recommended buffer size	All
✓	35.5%	26.4%	48.4%	54.4%
×	18.7%	8.1%	18.6%	6.4%

Results: Qmin Imperfections. Already De Vries *et al.* have shown that `qmin`-enabled resolvers send queries with three or more labels to the authoritative servers of the `.nl` TLD occasionally. The observations at `.nl` and `.se`, as shown in Fig. 8 and Fig. 9, confirm this finding. When neglecting queries to domain names and records for which `.nl` are authoritative (e.g., the domain names of the `.nl` name servers `[ns1-ns4].dns.nl`), we find that 77% of `qmin`-enabled resolvers send queries with more than two labels occasionally. In 55.1% of these

[9] https://bgp.tools/tags/dsl.csv.
[10] By setting the DO-flag in the query.
[11] As recommended by the 2020 DNS Flag day: https://www.dnsflagday.net/2020/.

cases the queries result in `NXDOMAIN` responses, signalling that the queried domain name does not exist. We could not find when exactly resolvers would fall back to sending the full query name, but this shows that even with `qmin` enabled, information about lower labels can leak. We could also observe this behaviour at resolvers of Google Public DNS and we reached out to their operators for clarification. Unfortunately, they could not explain to us what causes these occasional queries for fully-qualified domain names.

5 Controlled Experiments

The purpose of the controlled experiments is to look at the most recent versions of popular open source resolvers and look at how they handle minimized queries in regards to performance. In the controlled experiments by De Vries *et al.*, four open source resolvers were considered due to their popularity: Bind, Unbound, Knot Resolver, and PowerDNS. Only the first three resolvers had implemented `qmin` in their most recent version at the time, which meant that PowerDNS was excluded. PowerDNS has since then implemented and enabled relaxed `qmin` by default in version 4.3.0 and is therefore included in this study.[12] The versions of each resolver for the controlled experiments in this study were: Unbound 1.14.0, Bind 9.16.24, Knot Resolver 5.4.4 and PowerDNS 4.6.0. DNSSEC was turned off and the resolvers were configured to have the same size of caches.

5.1 Method

Just as in the original study, we use the Cisco Umbrella Top 1M list [5] for domains to query in the performance and error-rate measurements. The list contains the most popular queries based on passive DNS usage on their Umbrella global network. This list does not only contain browser-based HTTP user requests, but takes other protocols and non-end-users into account. Like in the study by De Vries *et al.*, domain names were aggregated from a timespan of two weeks (April 4th until April 19th, 2022) to avoid daily and weekly fluctuations and patterns. This resulted in 1.3M domain names with a mean of 3.26 labels, a median of 3 labels, a min of 1 and a max of 104 labels. This list of domains was sorted in four different orders to even out caching effects.

As mentioned in Sect. 2, there are two modes when enabling `qmin` on resolvers (i.e., relaxed and strict). These modes dictate whether the resolver should fall back to full query names when receiving `NXDOMAIN` or other unexpected responses from potentially broken name servers. For the controlled experiments in this study, both Unbound and Bind had the option to turn `qmin` off, run it in relaxed mode, or run it in strict mode. PowerDNS could either turn `qmin` off or turn it on in relaxed mode. Knot did not have any option to turn off `qmin` and only runs in relaxed mode. We used all the possible configurations in the experiments and compared the results to the results from the controlled experiments by De Vries *et al.*.

[12] https://doc.powerdns.com/recursor/settings.html#qname-minimization.

5.2 Results

As seen in Table 8, Unbound 1.14.0 is sending more packets compared to Unbound 1.8.0 when using `qmin` regardless of mode. The error-rate has decreased for all queries regardless of settings, and strict mode still have a higher error-rate than relaxed due to the extra `NXDOMAIN` responses from potentially broken name servers. Bind 9.16.24 is also sending out more packets compared to Bind 9.13.3 regardless of configuration.

Table 8. Resolver performance and result quality

Mode	#packets	Error
Unbound 1.8.0 (2018)		
Off	5.70M	12.6%
Strict	6.71M	15.9%
Relax	6.82M	12.6%
Knot 3.0.0 (2018)		
Relax	5.94M	13.5%
Bind 9.13.3 (2018)		
Off	5.07M	16.6%
Strict	5.84M	21.6%
Relax	6.39M	17.1%
Unbound 1.14.0 (2022)		
Off	5.93M	6.8%
Strict	8.50M	8.4%
Relax	8.68M	6.8%
Knot 5.4.4 (2022)		
Relax	3.14M	7.0%
Bind 9.16.24 (2022)		
Off	7.09M	11.2%
Strict	7.73M	11.2%
Relax	7.69M	11.2%
PowerDNS 4.6.0 (2022)		
Off	3.35M	7.0%
Relax	3.56M	7.0%

When looking closer at the traffic for Unbound and Bind we discovered that Unbound always resolves all name servers received and Bind is establishing TCP connections to the root servers. The error-rates for Bind have also decreased and there is no difference between strict and relaxed mode, which is unexpected. We have not been able to identify why relaxed mode showed no advantage. Knot is running relaxed `qmin` without any option to turn it off. The number of packets and the error-rate have decreased significantly in version 5.4.4 compared to version 3.0.0. The large decrease in number of packets is the opposite trend of Unbound and Bind. For PowerDNS the number of packets were similar when comparing relaxed `qmin` and no `qmin`, which was unexpected since `qmin` should produce more packets. Since PowerDNS enables `qmin` with the relaxed mode, the similar error-rates seen with and without `qmin` enabled was expected. This is true for Unbound and Bind as well. As the error-rates of all resolvers have decreased since 2018, regardless of `qmin` and mode, this could be a change on the name server side and not necessarily on the resolvers themselves.

The controlled experiments showed that the error-rate decreased for all resolvers compared to the original study, but the number of packets and the error-rate varied depending on the specific resolver and mode used. The `qmin` feature is tightly correlated with the number of packets, as a fully complete domain name typically requires fewer queries to resolve than a minimally built query that is iteratively resolved. Our results suggest that the performance of recursive DNS resolvers with `qmin` enabled has improved since the previous study, but further investigation is needed to fully understand the effects on each resolver. The number of packets can be an important factor in evaluating the performance of a resolver, as it can indicate the resources and communication required to process queries and retrieve responses. While a lower number of packets may indicate efficiency, other metrics such as latency and response accuracy should also be considered when assessing the performance of a resolver.

6 Discussion

In this section we summarize the results from our measurements and analyze the general adoption of `qmin`. Then we look at the improvements of the measurement methods as well as discuss the balance between performance and privacy.

6.1 Analysis of the Results

Looking at the summary of the measurement results compared to the results by De Vries *et al.* in Table 9 we can see that the active measurements (RIPE/open) show an increase in the adoption of `qmin`. The relatively high `qmin` share of RIPE suggest that the RIPE Atlas probes may be biased. People running RIPE Atlas probes in their network are likely technically adept and administrating resolvers with up-to-date security and privacy features. They may therefore not be representative of the average resolver on the Internet, but we see that the adoption of `qmin` has grown. The active measurements on the open resolvers paints perhaps a more accurate picture of `qmin`, but keep in mind that over 80% of the responding resolvers replied with an error, out of which over 90% are `REFUSED`. As mentioned earlier, this could be resolvers configured to only send queries on behalf of certain clients.

Table 9. Results of RIPE Atlas probe resolvers, open resolvers, K-Root and .nl ccTLD

Year	RIPE	Open	K-Root	.nl ccTLD
2018	10%	1.6%	0.6%	35.5%
2022	64%	16.42%	2.5%	57.3%

(.nl ccTLD numbers available are from 2019 and onwards.)

For the passive measurements at the root and the TLD we filtered out the invalid labels that affected the results of De Vries *et al.*. Here we also see a positive trend of minimized queries which matches the relative growth in the RIPE Atlas active measurements, a sixfold increase. The .nl TLD passive measurements also show an increase of `qmin`, but already in 2019 the share of incoming minimized queries was high. The resolvers querying domains at .nl are likely less representative of all DNS resolvers on the Internet, and instead point to the early adoption of privacy features in dutch DNS infrastructure. The results from the active and passive measurements show a clear and consistent increase in the adoption of `qmin`-enabled resolvers when comparing to the previous study. While the actual level of `qmin` adoption varies between the measurements, as they capture the behavior of different sets of resolvers from different vantage points, this is a positive development for Internet privacy.

6.2 Improvements of Measurements Methods

Our work builds heavily on the methods used by De Vries *et al.*, but also brings new insights based on enhancements to the methods. In the active measurements on open resolvers we observed that there were no significant differences between the three geographical locations, something that could not be inferred based on the measurements from a single location in the original study. The original study also classified open resolvers based on a single response. In this study we queried each open resolver 100 times, which revealed additional information when classifying the resolvers. In relation to improved measurement methods, we also set up a new domain using a wildcard label to mitigate cached delegations which was used when querying open resolvers in the second active measurements. DNSThought is currently not using a domain with a wildcard label, which means that some of the responses in the first active measurements could be cached false positives.

Some of the open resolvers were classified as conflicting resolvers in this study. With the use of a wildcard label we are able to mitigate any cached delegation happening, i.e., querying a non-minimizing resolver that has cached a recent minimized query from a `qmin`-enabled forwarding resolver. The most likely culprits are DNS load-balancers distributing query load across a pool of resolvers. According to Randall *et al.* [19], some DNS load-balancers could be using different software packages for their backend resolvers, which could be a cause of the conflicts that we observe. We also observed that 73% of the conflicting resolvers are responding with more NO than HOORAY, which is similar to the ratio of resolvers classified as `qmin`-enabled and not `qmin`-enabled.

6.3 Qmin Depth Limitation

The client-side active measurements at DNSThought are measuring `qmin` at the fourth-level domain. This means that the number of resolvers minimizing queries at lower levels (e.g., TLD and root) could be even higher. The Google Public DNS resolvers were classified as not `qmin`-enabled in the active measurements on open resolvers using a separate domain in Sect. 3.2. This was unexpected since the resolvers have been minimizing queries since 2020 according to DNSThought. Additional queries using different domains showed that the Google Public DNS resolvers were consistently responding differently based on domain. This was because Google wanted to get credit for minimizing queries at the root and TLD level, which originally did not show on DNSThought statistics.

Even though a resolver is not implementing `qmin` beyond the 2LD, a lack of data minimization within an authoritative DNS zone is less serious compared to fully disclosed query names at the root and TLD level. An organization registering a 2LD is most likely aware of their subdomains, so no harm would come from exposing those labels to their own name servers. Some organizations register domains under e.g., `.ac.uk` or `.co.jp` and it is therefore not as simple as to only minimize until the 2LD. We propose setting the depth limit using the Public Suffix List (PSL) [15] with one additional label (PSL+1). The PSL is a list maintained by Mozilla mainly used for restricting cookie setting. It contains effective TLDs (e.g., `.com` and `.net`) including those with more than one label (e.g., `.ac.uk` and `co.jp`.) A `qmin`-enabled resolver using the PSL+1 approach and looking up a RR for `a.b.example.ac.uk` should send `uk` to the root, `ac.uk` to the `.uk` ccTLD and then `example.ac.uk` to the name server of `.ac.uk`. The resolver would then stop minimizing and send `a.b.example.ac.uk` to the name server of `example.ac.uk` which is most likely the authoritative DNS zone. Since `ac.uk` is in the PSL, we refer to `example.ac.uk` as PSL+1.

7 Conclusion

We measured the adoption of query name minimization using active measurements from the client-side and passive measurements from the name server side. In addition we also performed controlled experiments on four open source resolvers to measure performance and error rate. We built the study on the methods of De Vries *et al.*, extended them, and included additional sets of data for the passive measurements. The extension of the methods includes measuring from multiple geographical locations and sending multiple queries to the resolvers instead of one. The latter revealed that some of the resolvers are sending both positive and negative responses, which was not observable with the previous method. The value of doing this replicated study comes from observing changes over time in the DNS ecosystem, improving the measurement methods and getting a picture of the `qmin` adoption shortly after the publication of RFC 9156, which builds on the result from the previous `qmin` adoption study.

The results of the active measurements of `qmin` adoption over time shows a positive trend with a rapid increase in 2020 when the resolvers in Google's ASN started minimizing queries. Plotting the share of `qmin`-enabled resolvers over time shows that 64% of the resolvers used by RIPE Atlas probes are minimizing queries. This is a significant adoption compared to 10% at the time of the previous study by De Vries *et al.*. The data used for the first active measurements with RIPE Atlas probes are displayed as part of DNSThought and our work has helped improve the probing for `qmin` adoption. With some communication with NLnet Labs and RIPE Atlas, a bug was fixed where new probes used by DNSThought were not querying for `qmin`. When looking closer at the Google Public DNS we observed that client-side active measurements using these resolvers seem to be highly dependent on specific domain names. With the help of NLnet Labs we found out that Google's resolvers have a `qmin` depth limitation, except for the domain used by DNSThought. This exception was done in order to get credit for minimizing at the root and TLD levels. In the controlled experiments using four open source resolvers we observed that the error-rates are decreasing. This is likely due to RFC 9156 which switched RR query types, specified fallbacks on certain errors, and added labels more dynamically and thus obsoleted the previous implementation of `qmin` in RFC 7816. But it could also be a change on the name server side. The number of packets are going up for two of the resolvers while it is decreasing or rather low for the other two, and it is still unclear why. We discussed the adequate level of minimizing query names in regards to both performance and privacy, where we argue that the privacy risks of leaking sensitive subdomains decrease after the authoritative DNS zone. We therefore look at the Public Suffix List as a possible resource for configuring the minimization depth limit.

Research Artifacts

To enable a third look at `qmin` in the future we provide whatever scripts used (https://github.com/Arcnilya/qmin2022) beyond what was already available by De Vries *et al.* [24]. The RIPE Atlas measurements by NLnet Labs are also accessible at RIPE [21].

Ethical Considerations

In this work we thought carefully about the ethical aspects of our measurements and disclosure. We used a list of open resolvers from third-party scans instead of doing the scan of the IPv4 address space on our own, thus avoiding adding more unnecessary load on the networks. We also spread out our active measurements in a round-robin style to not put too much load on single resolvers in a short span of time.

Acknowledgement. This work was founded by the Swedish Internet Foundation. The Day-in-the-live of the Internet dataset was kindly provided by DNS-OARC and the root server operators. We would like to thank Ulrich Wisser and Johan Stenstam for expertise within DNS, Roger Murray for help with querying .se ccTLD data, and Willem Toorop for understanding DNSThought and setting up the active measurements. We would also like to thank the reviewers and our shepherd Casey Deccio for valuable feedback.

References

1. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: DNS security introduction and requirements. RFC 4033, RFC Editor, March 2005. http://www.rfc-editor.org/rfc/rfc4033.txt
2. Bind: Bind documentation: options. https://bind9.readthedocs.io/en/v9_18_3/reference.html#options-statement-definition-and-usage. Accessed June 2022
3. Bortzmeyer, S.: DNS query name minimisation to improve privacy. RFC 7816, RFC Editorm March 2016
4. Bortzmeyer, S., Dolmans, R., Hoffman, P.: DNS query name minimisation to improve privacy. RFC 9156, RFC Editor, November 2021
5. Cisco: Cisco umbrella top 1m list. http://s3-us-west-1.amazonaws.com/umbrella-static/index.html. Accessed 12–25 Feb 2022
6. Cooper, A., et al.: Privacy considerations for internet protocols. RFC 6973, RFC Editor, July 2013
7. CZ.NIC: Measuring qname minimisation support. https://adam.pages.nic.cz/reports/adam/qname-minimisation-en/. Accessed Nov 2021
8. Google: Issue 1090985: Disable Intranet Redirect Detector by default. https://bugs.chromium.org/p/chromium/issues/detail?id=1090985. Accessed June 2020
9. Hoffman, P., McManus, P.: DNS queries over HTTPS (DoH). RFC 8484, RFC Editor, October 2018
10. Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., Hoffman, P.: Specification for DNS over transport layer security (tls). RFC 7858, RFC Editor, May 2016
11. Huitema, C., Dickinson, S., Mankin, A.: DNS over Dedicated QUIC Connections. RFC 9250m May 2022. https://doi.org/10.17487/RFC9250, https://www.rfc-editor.org/info/rfc9250
12. ICANN: M3: DNS root traffic analysis. https://ithi.research.icann.org/graph-m3.html. Accessed Mar 2022
13. Mockapetris, P.: Domain names - concepts and facilities. STD 13, RFC Editor, November 1987. http://www.rfc-editor.org/rfc/rfc1034.txt
14. Mockapetris, P.: Domain names - implementation and specification. STD 13, RFC Editor, November 1987. http://www.rfc-editor.org/rfc/rfc1035.txt
15. Mozilla Foundation: Public suffix list. https://publicsuffix.org/. Accessed 5 June 2008
16. NLnet Labs: DNSThought. https://dnsthought.nlnetlabs.nl/#qnamemin. Accessed 14 Oct 2018
17. NLnet Labs: Unbound documentation: qmin strict. https://unbound.docs.nlnetlabs.nl/en/latest/manpages/unbound.conf.html?highlight=relaxed%20qname#term-qname-minimisation-strict-yes-or-no. Accessed May 2021
18. Postel, J.: Internet protocol. STD 5, RFC Editor, September 1981. http://www.rfc-editor.org/rfc/rfc791.txt
19. Randall, A., et al.: Trufflehunter: cache snooping rare domains at large public DNS resolvers. In: Proceedings of the ACM Internet Measurement Conference, pp. 50–64 (2020)
20. Rapid7 Labs: UDP scans. https://opendata.rapid7.com/sonar.udp/. Accessed Jan 2022
21. RIPE Atlas: RIPE Atlas measurement. https://atlas.ripe.net/measurements/8310250/. Accessed 20 Apr 2017
22. Verisign: Chromium's Reduction of Root DNS Traffic. https://blog.verisign.com/domain-names/chromiums-reduction-of-root-dns-traffic/. Accessed Jan 2021

23. de Vries, W.B., Scheitle, Q., Müller, M., Toorop, W., Dolmans, R., van Rijswijk-Deij, R.: A first look at QNAME minimization in the domain name system. In: Choffnes, D., Barcellos, M. (eds.) PAM 2019. LNCS, vol. 11419, pp. 147–160. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-15986-3_10
24. de Vries, W.B., Scheitle, Q., Müller, M., Toorop, W., Dolmans, R., van Rijswijk-Deij, R.: A first look at qname minimization in the DNS, datasets. https://www.simpleweb.org/wiki/index.php/Traces#A_First_Look_at_QNAME_Minimization_in_the_Domain_Name_System. Accessed Oct 2022