A First Look at QNAME Minimization in the Domain Name System

Wouter B. de Vries[1], Quirin Scheitle[2], Moritz Müller[1,3], Willem Toorop[4], Ralph Dolmans[4], Roland van Rijswijk-Deij[1,4]

[1]University of Twente, [2]TUM, [3]SIDN Labs, [4]NLnet Labs

Abstract. The Domain Name System (DNS) is a critical part of network and Internet infrastructure; DNS lookups precede almost any user request. DNS lookups may contain private information about the sites and services a user contacts, which has spawned efforts to protect privacy of users, such as transport encryption through DNS-over-TLS or DNS-over-HTTPS. In this work, we provide a first look on the resolver-side technique of query name minimization (*qmin*), which was standardized in March 2016 as RFC 7816. *qmin* aims to only send minimal information to authoritative name servers, reducing the number of servers that full DNS query names are exposed to. Using passive and active measurements, we show a slow but steady adoption of *qmin* on the Internet, with a surprising variety in implementations of the standard. Using controlled experiments in a test-bed, we validate lookup behavior of various resolvers, and quantify that *qmin* both increases the number of DNS lookups by up to 26%, and also leads to up to 5% more failed lookups. We conclude our work with a discussion of *qmin*'s risks and benefits, and give advice for future use.

Keywords: DNS · Privacy · QNAME Minimization · Measurements

1 Introduction

The Domain Name System (DNS) plays a crucial role on the Internet. It is responsible for *resolving* domain names to IP addresses. The DNS is a hierarchical system where each so-called authoritative name server in the hierarchy is responsible for a part of a domain name. Recursive caching name servers – or 'resolvers' for short – query each level of authoritative name servers in turn to obtain the final answer. Resolvers usually cache responses to improve lookup speed.

On the Internet every domain resolution, given an empty cache, starts at the root of the DNS, which has knowledge of the name servers that are responsible for all the Top-Level Domains (TLDs). Those name servers typically then refer the recursive resolver on towards yet another name server. This can keep going indefinitely, only limited by the maximum query name (*qname*) length, until finally the authoritative name server for the requested *qname* is reached (in practice the recursive resolver can give up earlier).

In the standard DNS resolution process, outlined in RFC 1034 [24], the recursive resolver, unaware of zone cuts in which different parts of the domain are under control of different authorities, sends the full *qname* to each of the authoritative name servers in this chain. Since the first two (root and TLD) name servers in the recursion are very unlikely to be authoritative for the requested *qname*, this particular aspect causes unnecessary exposure of potentially private information [6]. E.g., exposing the *qname* of a website that is illegal in some countries to more parties than necessary might put the querying end-user at serious risk. A solution for this issue is proposed in RFC 7816 [7], which introduces query name minimization (*qmin*), preventing recursive resolvers from sending the full *qname* until the authoritative name server for that *qname* is reached [7].

End-users typically do not run a recursive resolver, but instead depend on others, such as their ISP, to enable this privacy-preserving feature. From a user's perspective, *qmin* is difficult to detect, making it hard to judge adoption.

In this paper we study the adoption, performance, and security implications of RFC 7816. Specifically, we: **1)** develop novel methodology to detect whether a resolver has *qmin* enabled, and quantify the adoption of *qmin* over time, both with active measurements from the end-user perspective, and passive measurements from the authoritative name server perspective, at a root and TLD server, **2)** develop an algorithm to fingerprint *qmin* implementations, and classify the use of *qmin* algorithms in the Internet and, **3)** provide insight into the impact of *qmin* on performance and result quality for three resolver implementations.

In order to facilitate reproducibility we make our scripts and datasets available publicly [33].

2 Background and Related Work

When DNS was first introduced in the 1980s, there was no consideration for security and privacy. These topics have now gained considerable importance, leading to a plethora of RFCs that add security and privacy to the DNS. For example, DNSSEC [28–30] introduces end-to-end authenticity and integrity, but no privacy. More recently, DNS-over-TLS [21] and DNS-over-HTTPS [20] added transport security. "Aggressive Use of DNSSEC-Validated Cache" [18], reduces unnecessary leaks of non-existing domain names. Furthermore, running a local copy of the root zone at a resolver avoids sending queries to root servers completely [19].

Typically, resolvers send the full *qname* to each authoritative name server involved in a lookup. Consequently, root servers receive the same query as the final authoritative name server. Since the IETF states that Internet protocols should minimize the data used to what is necessary to perform a task [12], *qmin* was introduced to bring an end to this. Resolvers that implement *qmin* only query name servers with a name stripped to one label more than what that name server is known to be authoritative for. E.g., when querying for *a.b.domain.example*, the resolver will first query the root for *.example*, instead of *a.b.domain.example*. The reference algorithm for *qmin* also hides the original query type by using the `NS` type instead of the original until the last query. In Table 1 we show what queries are performed for both standard DNS and the *qmin* reference implementation.

Table 1: DNS queries and responses without (left) and with (right) *qmin*.

Standard DNS resolution			*qmin* Reference (RFC 7816)		
a.b.example.com.	A	→ .	com.	NS	→ .
com.	*NS*	← *.*	*com.*	*NS*	← *.*
a.b.example.com	A	→ com.	example.com	NS	→ com.
example.com	*NS*	← *com.*	*example.com*	*NS*	← *com.*
a.b.example.com	A	→ example.com.	b.example.com	NS	→ example.com.
a.b.example.com	*A*	← *example.com.*	*b.example.com*	*NS*	← *example.com*
			a.b.example.com	NS	→ example.com.
			a.b.example.com	*NS*	← *example.com*
			a.b.example.com	A	→ example.com.
			a.b.example.com	*A*	← *example.com*

This reference algorithm, however, faces two challenges on the real Internet: First, it does not handle configuration errors in the DNS well [26]. E.g., in case *b.domain.example* does not have any RRs but *a.b.domain.example* does, a name server should respond with `NOERROR` for a query to *b.domain.example* [8], but in fact often responds with `NXDOMAIN`, or another invalid `RCODE`. This would force resolvers that conform to the standard to stop querying and thereby not successfully resolve the query. Also, operators report other issues, such as name servers that do not respond to `NS` queries, which would break *qmin* as well [25].

Second, *qmin* can lead to a large number of queries. For example, a name with 20 labels would make the resolver issue 21 queries to authoritative name servers, causing excessive load at the resolver and authoritative. Attackers can abuse this for DoS attacks by querying excessively long names for victim domains.
Both of these issues led resolver implementors to modify their *qmin* implementations, as well as adding so called “strict” and “relaxed” modes, which we investigate in Subsection 3.2 and Section 5.

As of October 2018, three major DNS resolvers support *qmin*. Unbound supports *qmin* since late 2015 and turned relaxed *qmin* on by default in May 2018 [25]. Knot resolver uses relaxed *qmin* since its initial release in May 2016 [13], and the recursive resolver of BIND supports *qmin* and turned the relaxed mode on by default in July 2018 [23]. Another frequently used resolver, PowerDNS Recursor, does not support *qmin* yet [9].

Related Work: Hardaker et al. [19] showed that root servers receive a considerable amount of privacy-sensitive query names, and propose using local instances of root servers to alleviate this issue. Imana et al. [22] study this aspect from a broader perspective, covering all name servers above the recursive resolver, and report similar privacy issues.

Schmitt et al. [32] propose Oblivious DNS, an obfuscation method introducing an additional intermediate resolver between recursive resolver and authoritative name servers. Oblivious DNS prevents the additional resolver from learning the user’s IP address and the recursive resolver from learning the query name.

Recent work [34] has also shown that *qmin* increases the number of queries per lookup, increasing the load on authoritative name servers. They provide a technique called `NXDOMAIN` optimization that reduces the number of queries in case the resolver encounters an `NXDOMAIN`. We extend this by providing longitudinal measurements, showing various implementations of *qmin* algorithms and quantifying the increase in queries per resolver implementation.

3 Active Internet-Wide Measurements

We conduct active Internet-wide measurements using two methods. First, we use RIPE Atlas probes to query a domain under our control. Second, we query open resolvers for the same domain. RIPE Atlas is a global measurement network with over 10,000 small devices called probes, and 370 larger probes, called anchors. In this section, we measure *qmin* adoption over time, classify the various *qmin* implementations in use, and shed light on *qmin* use by open resolvers.

3.1 Resolver Adoption over Time

We detect *qmin* support by relying on the fact that a non-*qmin* resolver will miss any delegation that happens in one of the labels before the terminal label. So, if we delegate to a different name server, with a different record for the terminal label in one of the labels *before* the terminal label, *qmin* resolvers will find a different answer than non-*qmin* resolvers.

We scheduled a RIPE Atlas measurement for all probes to perform a lookup with all the probe's resolvers for *"a.b.qnamemin-test.domain.example"* with type `TXT` [1], repeating every hour. Each probe uses its own list of resolvers, typically obtained via DHCP, and assumed typical for the network that hosts the probe.

A non-*qmin* resolver will send a query for the full qname to the authoritative name server for "qnamemin-test.domain.example", and will end up with a `TXT` reply containing the text: "`qmin NOT enabled`." A *qmin* resolver will send a query for just the second-to-last label, "b.qnamemin-test.domain.example", to the authoritative name server for "qnamemin-test.domain.example". For this minimized query, it will receive a delegation to a different name server, which will return a `TXT` record containing the text: "`qmin enabled`."

This measurement runs since April 2017, and allows us to see the long term adoption of *qmin*. Figure 1b shows the overall adoption of *qmin* as seen from all RIPE Atlas probes. We count both probes and probe/resolver combinations, as a significant number of probes uses multiple resolvers. Adoption grew from 0.7% (116 of 17,663) of probe/resolver pairs in April 2017 to 8.8% (1,662 of 18,885) in October 2018. Also in April 2017, 0.9% (82 of 9,611) of RIPE Atlas probes had at least one *qmin* resolver, growing to 11.7% (1,175 of 10,020) in October 2018.

In Figure 1a only probe/resolver pairs supporting *qmin* are shown. We see a steep rise of *qmin* resolvers in April 2018. Figure 1a also shows probes that have at least one *qmin* resolver as well as at least one resolver that does not do *qmin*.

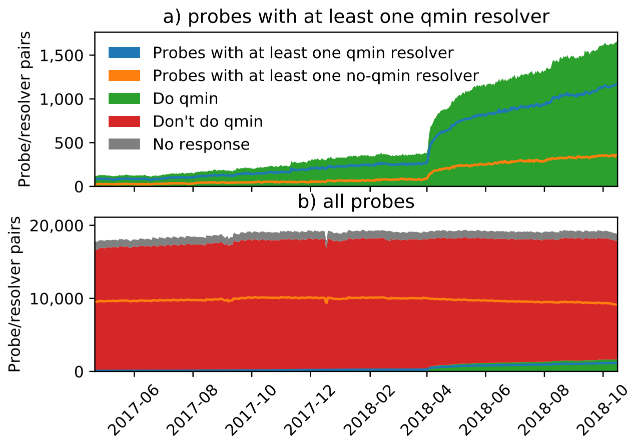

Fig. 1: Adoption over Time

It is noteworthy that at the last measurement (October 15, 2018) at least 31% of probes that have a *qmin* resolver, also have at least one non-*qmin* resolver.

Alongside the *qmin* measurement, we run measurements that return the IP address of the resolver as seen from an authoritative name server [2, 3, 5]. By identifying the Autonomous System Numbers (ASNs) associated with the IP addresses seen at the authoritative name server we gain insight in the organizations providing the *qmin* resolvers. From this we learn that the adoption of Cloudflare (`1.1.1.1`) is responsible for the fast rise of *qmin* resolvers in April 2018.

We also found some public resolvers, such as Google Public DNS, that in some cases appear to support *qmin* according to our test, but in fact do not. This is likely caused by a *qmin*-enabled *forwarding* resolver, which forwards to, in Google's case, `8.8.8.8`. Additionally, the non-*qmin* resolver successively caches the authoritative for the second-last label and will appear to support *qmin* for the TTL of the delegation (10 seconds in our test). We have developed an improved test without these issues in the course of this research, but this corrected test did not yet exist during scheduling of the RIPE Atlas measurement in April 2017.

The improved test, "a.b.*random-element*.domain.example. `TXT`", uses a random pattern as the third-last label which is uniquely chosen for each query, preventing other measurement queries to find a cached delegation for the second-last label. This improved test is used in measuring the adoption by open resolvers in Subsection 3.3, removing false positives from that measurement.

We argue that this flaw had little impact on our results, as *i)*, RIPE Atlas measurements are spread out over an hour, whereas our test record has a small TTL, reducing this risk and *ii)* the overall trend over time is still indicative.

The ASNs seen at the authoritative were further used to classify resolvers in three categories: **1)** *Internal* resolvers have the same ASN for the probe and the observed resolver IP, **2)** *External* resolvers for which the ASN of the resolver IP configured on the probe matches the ASN for the IP observed on the authoritative, but differs from the ASN in which the probe resides, **3)** *Forwarding* resolvers, for which the ASN seen on the authoritative differs from both the ASN associated with the resolver IP configured on the probe and the ASN the probe resides in.

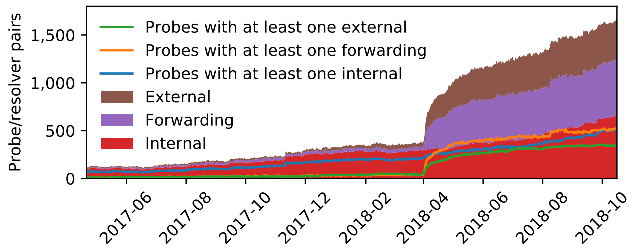

Fig. 2: Internal, Forwarding and External resolvers supporting *qmin*

Figure 2 shows that both *External* and *Forwarding* probe/resolver pairs supporting *qmin* are on the rise, which is mainly due to adoption of the Cloudflare resolver in April 2018. We can also see that *qmin* support is steadily growing with *Internal* resolvers, which do not include the larger public resolvers.

Looking more closely at the *Internal* resolvers we have identified, we see that several ISPs started supporting *qmin* over the past 1.5 years. Most notably "Versatel Deutschland GmbH" started supporting *qmin* on November 9th, 2017; "Init Seven AG" on August 2nd, 2017; "OVH Systems" on February 1st, 2018; and "M-Net Telekommunikations GmbH, Germany" on May 1st, 2018. Note that these do not necessarily cause a visible change in Figure 2.

3.2 Fingerprinting Resolver Algorithms

As described in Section 2, the RFC [7] provides a reference algorithm for *qmin*. This is an aggressive algorithm in the sense that it maximizes potential privacy gains at the cost of performance. It iteratively increases the name length by one label, querying for the NS type, until it reaches the full name. Then, it switches to the original query type, thus also this type from all but the final name server.

While this algorithm is good for privacy, it can significantly impact performance, security, and result quality (see Section 5). Since the reference algorithm is merely a suggestion, resolver implementors are free to write their own algorithm.

Using RIPE Atlas measurements, we explore *qmin* algorithms implemented in practice. To measure this, we performed a one-off DNS measurement [4] from all RIPE Atlas probes able to resolve `A` records correctly (9,410 probes). We control the authoritative name server for the queried name, permitting us to identify query behavior. The queried name consists of 24 labels, including random values and the probe ID to permit mapping inbound DNS queries to originating probes. We see inbound queries from 8,894 unique probes (out of 9,410) from 8,179 unique

Table 2: Top 6 signatures seen at Authoritative Resolvers, mapped to resolver implementations. Reference implementation not observed.

Type	**Signature**	**Implementation**	**Count**
1	24A		13,892
2	3NS-24A	Knot 3.0.0	784
3	3A-4A-5A-8A-11A-14A-17A-21A-24A		239
3	3A-4A-5A-6A-9A-12A-15A-18A-22A-24A		193
3	3A-4A-7A-10A-13A-16A-20A-24A	Unbound 1.8.0	16
4	3NS-4NS-5NS-24A	BIND 9.13.3	11
	3NS-4NS-5NS-6NS-7NS-...-24NS-24A	Reference	0

resolvers. Most probes have at least two resolvers configured, many overlapping with those of other probes, resulting in 20,716 total inbound queries.

Assigning Signatures: To group resolver behavior, we map the incoming query behavior observed at our authoritative name server to signatures, containing length, order, and type of inbound queries. Our test domain is at the second label depth, so we observe queries starting from the third label depth. For example, an algorithm asking for `NS` at the 3rd label, then for `NS` at the 4th label, and then for `A` at the final, 24th, label, will be mapped to the signature `3NS-4NS-24A`.

Signatures of BIND, Knot and Unbound: To have a basis for comparison, we run our domain through each of these three resolvers, which are known to implement *qmin*, and determine each of their *qmin* signatures. BIND and Unbound also support an additional *strict mode*, however, this has no effect on the signature and is related to how `NXDOMAIN` responses are handled. The resulting signatures, and the reference algorithm signature, are shown in Table 2.

Signatures in the Wild: We identify four types of signatures, with some types having multiple variations, see Table 2. The first, most common type (*#1*) applies no *qmin*. These resolvers directly query the full length DNS name. The second type (*#2*) is a minimalistic *qmin* approach. After a no-delegation check below the base domain, the full query name is sent. This is used by the Knot resolver, and, for example, by Cloudflare's public DNS resolver. The third type, with variations (*#3*), is closer to the reference algorithm, but displays various ways of skipping labels, as well as always using the `A` query type instead of the `NS` type as suggested by the reference algorithm. Unbound is known to have a similar implementation [16], confirmed in our experiments. The final signature, (*#4*) uses the NS query type, and jumps to querying for the full name after not finding a zone cut for three labels. This is consistent with the BIND implementation.

Besides the specific signatures seen in Table 2, there are many variations of type #3. This indicates that not only do different resolvers implement different algorithms, but they also appear to be configurable or change over time (e.g. a new version changes the behavior). In total we see 20 different signatures, many of which only from one specific resolver. Interestingly, we did not observe the reference algorithm from any resolver.

3.3 Adoption by Open Resolvers

Aside from resolvers that can be reached from inside networks, such as those offered by ISPs, there are also a large number of open resolvers on the Internet. These can range from unsecured corporate DNS resolvers, to large scale public DNS services, such as those run by Google, OpenDNS, Quad9 and Cloudflare.

Rapid7 provides a list of servers that are responsive on UDP port 53, which are typically DNS servers. We query each such server using the method outlined in Subsection 3.1. The list contains a total of 8M IPv4 addresses, we receive a response from 64% of these. Of those responding, 32% respond with a NOERROR reply, of which only 72% (≈1.2M) provide a correct reply.

Of those 1.2M, only 19,717 (1.6%) resolvers support *qmin*. On the authoritative side, we only observe 110k unique source IPs, which suggests that many of the queried resolvers are in fact forwarders. Of the resolvers that implement *qmin*, 10,338 send queries from a Cloudflare IP, 2,147 from an OVH IP, and 1,616 from a TV Cabo Angola IP address. This shows that most *qmin*-supporting open resolvers simply forward to larger public DNS resolvers that implement *qmin*.

For *qmin*-enabled resolvers, we compare the ASN of the IP we send our query to with the ASN of the IP seen at the authoritative for that same query. We find 11.5k resolvers to resolve externally, and 8.2k resolvers to resolve internally.

The takeaway is that many open resolvers on the Internet use centralized public DNS services. Thus, efforts to drive adoption of *qmin* should focus on large public DNS providers (e.g. Google, which does not support *qmin* yet).

4 Passive Measurements at Authoritative Name Servers

As *qmin* limits the visible information of a query at authoritative name servers, adoption of *qmin* likely changes the query profile of resolvers as observed on the authoritative side. We measure the impact and adoption of *qmin* with query data collected at the authoritative name servers of the ccTLD *.nl* and of K-Root.

Name servers of *.nl* are authoritative for the delegation of 5.8 million domain names. If they receive queries for a *.nl* domain name with 2 or more labels then they almost always (except for DS records) respond with a set of name servers that are actually responsible for the queried domain name. Thus, a query for the NS record of a second level domain name is sufficient for the *.nl* name servers to answer the query. Similarly, the root servers are authoritative for the 1.5k TLDs as of October 9, 2018, and a query for just the TLD is sufficient in most cases.

We cannot be certain whether resolvers send minimized queries to the authoritative name servers, but we can count the queries that follow the expected patterns if resolvers were to send minimized queries. For the rest of this section, and following the observations made in Section 3, we count queries as minimized if the query contains only 2 labels (at *.nl*) or 1 label (at K-Root). With increasing *qmin* adoption, we expect to see an increase in queries that follow these criteria.

Identifying qmin First, we measure how query patterns seen at the authoritative name servers differ when resolvers implement *qmin*. We use the list of open resolvers from Subsection 3.3 of which we know whether they have *qmin* enabled. Then, we count how many queries these resolvers send to the authoritative name servers of *.nl* for names with just two labels on 2018-10-11. In total, we observe 1,918 resolvers that *do* and 27,251 resolvers that *do not* support *qmin*.

In Figure 3 we see that *qmin*-enabled resolvers send a median of 97% of queries classified as minimized, whereas resolvers that have not enabled this feature send only 12% of their queries classified as minimized. This confirms that *qmin* has an observable impact at authoritative name servers.

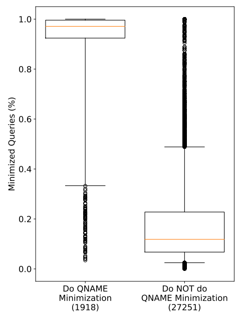

Fig. 3: Minimized queries to *.nl*. Whiskers at 0.05 and 0.95.

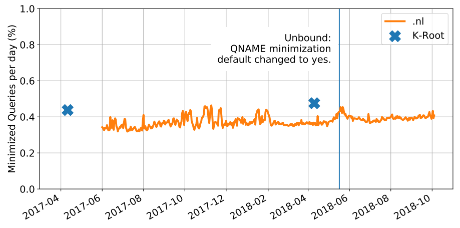

Fig. 4: Share of minimized queries sent to *.nl* and K-Root

Resolver Adoption over Time Based on the results of the previous section we expect a visible impact from increasing adoption of *qmin* at authoritative name servers. To verify this expectation we count how many queries overall are sent for 2nd level domain names and TLDs respectively. We analyze *.nl* data collected from 2017-06-01 to 2018-09-30 at 2 of the 4 authoritative name servers [35] and rely on the "Day In The Life of the Internet" (DITL) data sets of K-Root on 2017-04-11 and 2018-04-10 collected by DNS-OARC [15]. We observe more than 400B queries from 2017-06-01 to 2018-09-30 at *.nl* and 12B queries on the two days of the DITL data sets. Figure 4 shows the fraction of minimized queries.

In the beginning of our measurement, roughly 33% of the queries to *.nl* where minimized. A year later, at least 40% of queries were minimized. A peak around May 2018 correlates with the date on which Unbound enabled *qmin* by default. This peak, however, is followed by a steep decline shortly after, which means we cannot confirm if Unbound enabling *qmin* by default caused this peak.

At K-Root we also observe an increase from 44% to 48% in queries for domain names with only one label. Note that query patterns at the root may strongly vary from one day to another and that many queries are sent to non existing domain names which can influence our results [10].

Table 3: Performance and result quality across *qmin* modes and resolvers. Results are mean (μ) across all runs, with all standard deviations $\sigma < 2\%\mu$. We also show the *qmin* algorithm signature per resolver for the *qmin*-enabled case (signature without *qmin* is always `24A`).

	Unbound 1.8.0			Knot 3.0.0	Bind 13.3.2		
qmin Signature	`3A-4A-7A-...-24A`			`3NS-24A`	`3NS-4NS-5NS-24A`		
qmin mode	off	relaxed	strict	relaxed	off	relaxed	strict
# packets	5.70M	6.82M	6.71M	5.94M	5.07M	6.39M	5.84M
Errors	12.6%	12.6%	15.9%	13.5%	16.6%	17.1%	21.6%

5 Controlled Experiments: Impact on Resolver Performance and Result Quality

As *qmin* is deployed at the recursive resolver, we explore how *qmin* impacts the *performance* and the *result quality* of such a recursive resolver. We compare three popular *qmin*-enabled resolvers in their most recent version: Unbound 1.8.0, Knot 3.0.0, and BIND 9.13.3. We use all three resolvers with their default options, only adjusting to an equal cache size of 4GB and turning DNSSEC validation off[1]. We cycle through all configurable *qmin* behaviors for Unbound and BIND; Knot has *relaxed qmin* hardcoded. As target domains, we use the Cisco Umbrella Top 1M [11] list as a sample of popular domain names, and aggregate all domains names for a 2-week period to avoid daily fluctuations and weekly patterns [31], resulting in 1.56M domain names. To even out caching effects, we sort our target domain names in 4 different orders. We conduct several iterations of these measurements from October 1 through October 15, 2018, starting each measurement with an empty cache. We report means from all measurement runs, and find little variation in all numbers, typically one standard deviation σ is smaller than 2% of the mean μ. Table 3 gives an overview of our results.

Performance: *qmin* shows a clear impact on the number of packets sent to resolve our 1.56M domains. For Unbound, the 5.7M packets without *qmin* require 6.82M (relaxed) and 6.71M (strict) packets with *qmin*, a 17–19% increase. For BIND, the increase is 15–26%. It is to be expected that the strict mode requires fewer packets, as it will give up on receiving an error, whereas relaxed modes continue through `SERVFAIL` or `NXDOMAIN` error codes. This increase in packet count is not offset by smaller packets, across resolvers we see average packet sizes only decrease by 5% or less with *qmin* enabled.

This confirms that *qmin* in its current form does come with a performance penalty of up to 26%. We argue that the full cache in a production resolver will soften that overhead. Please note that a comparison of packet counts between different resolvers implicitly compares many other details such as caching strategies,

[1] We turn DNSSEC validation off to achieve comparable behavior (validating DNSSEC requires more queries to be sent); we also note that the combination of *qmin* and DNSSEC may induce further complexities beyond the scope of this work.

which is why comparison between resolvers should be conducted very carefully. While it may seem intuitive that Unbound's `3A-4A-7A-10A-13A-16A-20A-24A` *qmin* approach requires more packets than Knot's `3NS-24A` and BIND's `3NS-4NS--5NS-24A` approaches (cf. Subsection 3.2), a comparison of the number of packets between resolvers would require a much deeper exploration of root causes of packets sent, such as caching and time-out strategies.

Result quality: Another critical aspect of resolver performance is the result quality: Will a resolver be able to work through numerous edge cases and misconfigurations to deliver a response, or will it hang up on certain errors? To answer this question, we compare the amount of errors (NXDOMAIN or SERVFAIL) in our resolution results between different resolver and *qmin* approaches. Across resolvers, we see a significantly higher share of errors with strict *qmin* enabled. For example, the 3.3% increase for Unbound translates to ≈50k domains, a significant share of these popular DNS domain names. The difference in resolvers corresponds to our observations on resolver behavior: As reported in Section 2, a portion of authoritative name servers fails to respond to NS queries. As Unbound uses type A queries to discover zone boundaries, and Knot and BIND use NS queries (as suggested by RFC 7816), higher error rates are expected for Knot and BIND. The surprisingly high baseline of non-resolving domains of 12-16% is a characteristic of the Umbrella Top 1M list recently discussed in [31].

These findings show that *qmin* comes with two drawbacks: Packets and bytes transferred increase, and, depending on the detailed algorithm, also a significant share of popular DNS names fails to resolve.

6 Discussion and Conclusions

Our study covered *qmin* from various angles: we performed **1)** controlled experiments that confirm that *qmin* can have negative performance and result quality implications, and **2)** active and passive measurements in the Internet that confirm from both the client and authoritative server side that *qmin* adoption is rising. We also explored the various problems and workarounds that have been deployed, and want to conclude and discuss further aspects:

***qmin* is complex:** Like many DNS mechanisms, *qmin* sounds simple, but broken deployments make it difficult to implement without collateral damage. Resolvers' iterations towards a relaxed *qmin* algorithm reflect this, and important take-aways are: *(i)* Using NS queries to detect zone cuts results in a considerable number of failures; using A queries instead seems reasonable. *(ii)* responding to SERVFAIL/NXDOMAIN by sending the full name (*i.e.,* disabling *qmin* for this query) is currently a necessity to avoid significant error rates.

***qmin* can be a security risk:** Having a resolver step through many iterations for a name with an excessive number of labels is a DoS attack vector. All implementations we encountered mitigate this. Unbound jumps over labels to decrease the number of queries to some maximum, considerably saving on query count. Knot's (`3NS-24A`) and BIND's (`3NS-4NS-5NS-24A`) approaches go further: Knot stops *qmin* if it encounters a label that has not been delegated (except for some exceptions, such as *.co.uk*). BIND has both a limit on the maximum number of labels (default 9), in addition to having a maximum number of undelegated labels (default 3). We consider these approaches good, as they mitigate security risks while still providing *qmin* privacy against the top levels in the DNS hierarchy.

***qmin* can impact resolver performance and result quality:** Currently, *qmin* comes with a 15%+ performance penalty, and unless implemented very carefully, will also impair result quality. Please note that, as *qmin* queries are sent sequentially, the measured increase in query volume will correlate to latency.

Recommendations: Based on the insights collected in this paper, we conclude with the following recommendations: ***(i)***, despite its performance and quality caveats, *qmin* improves privacy and should be universally deployed. ***(ii)*** *qmin* deployment must be conducted carefully: We recommend an algorithm that combines Unbound's and BIND's algorithms, *i.e.,* conducts fallback upon error, replaces NS (and other) query types by A queries, and stops *qmin* after a configurable number of labels.***(iii)*** over time, heuristics may be added to alleviate certain cases where *qmin* will unlikely add privacy. For example, DANE-TLSA labels such as `_443._tcp` could be exempt from *qmin*.

Conclusion: The currently still rather low *qmin* adoption already causes a significant positive effect for query privacy at both Root and TLD authoritative name servers. While there are legitimate performance, result quality, and security concerns, we already see resolver implementers tackle these, and are confident that these negative implications will be further reduced, assisted by the quantitative evidence and tangible recommendations in this study. We fully expect more and more DNS operators to enable *qmin* to further improve privacy of end-users on the Internet.

Ethical Considerations and Reproducibility: We carefully considered ethical implications of our work. We followed scanning best practices [17], and received no complaints. Our passive data collection has been cleared by the respective IRBs, and we follow the recommendations by Dittrich *et al.* [14] and Partridge *et al.* [27]. To encourage other researchers to validate and/or build upon our results we publish our scripts, code and data publicly [33].

Acknowledgements: This work was partially funded by the German Federal Ministry of Education and Research under project X-Check (grant 16KIS0530). Partial funding was also supplied by SURFnet Research on Networks.

References

1. RIPE Atlas measurement for `a.b.qnamemin-test.internet.nl TXT`. `https://atlas.ripe.net/measurements/8310250/` (2017)
2. RIPE Atlas measurement for `o-o.myaddr.l.google.com TXT`. `https://atlas.ripe.net/measurements/8310237/` (2017)
3. RIPE Atlas measurement for `ripe-hackathon6.nlnetlabs.nl AAAA`. `https://atlas.ripe.net/measurements/8310366/` (2017)
4. RIPE Atlas measurement for `ripe-hackathon6.nlnetlabs.nl AAAA`. Ripe MSM IDs: 16428213, 16428214, 16428215, 16428216, 16428217, 16428218, 16428219, 16428220, 16428221, 16428222 (2017)

5. RIPE Atlas measurement for `whoami.akamai.net A`. `https://atlas.ripe.net/measurements/8310245/` (2017)
6. Bortzmeyer, S.: DNS Privacy Considerations. RFC 7626 (Informational) (Aug 2015), `https://www.rfc-editor.org/rfc/rfc7626.txt`
7. Bortzmeyer, S.: DNS Query Name Minimisation to Improve Privacy. RFC 7816 (Experimental) (Mar 2016), `https://www.rfc-editor.org/rfc/rfc7816.txt`
8. Bortzmeyer, S., Huque, S.: NXDOMAIN: There Really Is Nothing Underneath. RFC 8020 (Proposed Standard) (Nov 2016), `https://www.rfc-editor.org/rfc/rfc8020.txt`
9. Bortzmeyer, S.: PowerDNS - Add Qname minimisation (2015), `https://github.com/PowerDNS/pdns/issues/2311`
10. Castro, S., Wessels, D., Fomenkov, M., Claffy, K.: A Day at the Root of the Internet. ACM SIGCOMM Computer Communication Review **38**(5), 41–46 (2008)
11. Cisco: Cisco Umbrella Top 1M List. `https://s3-us-west-1.amazonaws.com/umbrella-static/index.html`, Sep 14 – Sep 30, 2018
12. Cooper, A., Tschofenig, H., Ph.D., D.B.D.A., Peterson, J., Morris, J.B., Hansen, M., Smith, R.: Privacy Considerations for Internet Protocols. RFC 6973 (Jul 2013), `https://rfc-editor.org/rfc/rfc6973.txt`
13. CZ.NIC: Knot resolver 1.0.0 released (2016), `https://www.knot-resolver.cz/2016-05-30-knot-resolver-1.0.0.html`
14. Dittrich, D., Kenneally, E., et al.: The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. US Department of Homeland Security (2012)
15. DNS OARC: Day In The Life of the Internet (2017 and 2018), `https://www.dns-oarc.net/oarc/data/ditl`
16. Dolmans, R.: QNAME Minimization in Unbound. `https://ripe72.ripe.net/wp-content/uploads/presentations/120-unbound_qnamemin_ripe72.pdf` (2016), RIPE 72
17. Durumeric, Z., Wustrow, E., Halderman, J.A.: ZMap: Fast Internet-wide Scanning and Its Security Applications. In: USENIX Security (2013)
18. Fujiwara, K., Kato, A., Kumari, W.: Aggressive Use of DNSSEC-Validated Cache. RFC 8198 (Proposed Standard) (Jul 2017), `https://www.rfc-editor.org/rfc/rfc8198.txt`
19. Hardaker, W.: Analyzing and Mitigating Privacy with the DNS Root Service. In: NDSS: DNS Privacy Workshop, 2018 (2018)
20. Hoffman, P.E., McManus, P.: DNS Queries over HTTPS (DoH). RFC 8484 (Oct 2018), `https://rfc-editor.org/rfc/rfc8484.txt`
21. Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., Hoffman, P.E.: Specification for DNS over Transport Layer Security (TLS). RFC 7858 (May 2016), `https://rfc-editor.org/rfc/rfc7858.txt`
22. Imana, B., Korolova, A., Heidemann, J.: Enumerating Privacy Leaks in DNS Data Collected Above the Recursive. In: NDSS: DNS Privacy Workshop, 2018. San Diego, California, USA (feb 2018), `https://www.isi.edu/%7ejohnh/PAPERS/Imana18a.html`
23. ISC: Release notes for bind version 9.13.2 (2018), `https://ftp.isc.org/isc/bind9/9.13.2/RELEASE-NOTES-bind-9.13.2.txt`
24. Mockapetris, P.: Domain names - concepts and facilities. RFC 1034 (Nov 1987), `https://rfc-editor.org/rfc/rfc1034.txt`
25. NLnet Labs: Nlnet labs: Unbound chanelog (2018), `https://nlnetlabs.nl/svn/unbound/tags/release-1.8.0/doc/Changelog`

26. Pappas, V., Wessels, D., Massey, D., Lu, S., Terzis, A., Zhang, L.: Impact of Configuration Errors on DNS Robustness. IEEE Journal on Selected Areas in Communications **27**(3), 275–290 (2009)
27. Partridge, C., Allman, M.: Ethical Considerations in Network Measurement Papers. Communications of the ACM (2016)
28. Rose, S., Larson, M., Massey, D., Austein, R., Arends, R.: DNS Security Introduction and Requirements. RFC 4033 (Mar 2005), `https://rfc-editor.org/rfc/rfc4033.txt`
29. Rose, S., Larson, M., Massey, D., Austein, R., Arends, R.: Protocol Modifications for the DNS Security Extensions. RFC 4035 (Mar 2005), `https://rfc-editor.org/rfc/rfc4035.txt`
30. Rose, S., Larson, M., Massey, D., Austein, R., Arends, R.: Resource Records for the DNS Security Extensions. RFC 4034 (Mar 2005), `https://rfc-editor.org/rfc/rfc4034.txt`
31. Scheitle, Q., Hohlfeld, O., Gamba, J., Jelten, J., Zimmermann, T., Strowes, S.D., Vallina-Rodriguez, N.: A Long Way to the Top: Significance, Structure, and Stability of Internet Top Lists. In: IMC'18, arXiv:1805.11506. Boston, USA (Nov 2018)
32. Schmitt, P., Edmundson, A., Feamster, N.: Oblivious DNS: Practical Privacy for DNS Queries. arXiv:1806.00276 (2018)
33. de Vries, W.B., Scheitle, Q., Müller, M., Toorop, W., Dolmans, R., van Rijswijk-Deij, R.: Datasets and Scripts (2019), `https://www.simpleweb.org/wiki/index.php/Traces#A_First_Look_at_QNAME_Minimization_in_the_Domain_Name_System`
34. Wang, Z.: Understanding the Performance and Challenges of DNS Query Name Minimization. In: 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). pp. 1115–1120. IEEE (2018)
35. Wullink, M., Moura, G.C., Müller, M., Hesselman, C.: ENTRADA: A High-Performance Network Traffic Data Streaming Warehouse. In: Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP. pp. 913–918. IEEE (2016)