Understanding the Performance and Challenges of DNS Query Name Minimization

Zheng Wang
National Institute of Standards and Technology, MD 20899, USA
Email: zhengwang98@gmail.com

***Abstract*—As a promising solution to DNS privacy, query name minimization limits the unnecessary leakage of query name information in DNS requests. Due to the lack of detailed measurement study, there is little understanding of the performance, compatibility, and security implications of query name minimization. In this paper, we measure the performance of query name minimization. We find that query name minimization requires a significant query increase but NXDOMAIN optimization technique can alleviate the increase. We propose a DDoS vulnerability with query name minimization and evaluate its impacts and defenses using NXDOMAIN optimization. Broken empty non-terminals are measured and identified as the critical obstacles to the transition to query name minimization.**

I. Introduction

The Domain Name System (DNS) is an indispensable large-scale database for the today's Internet. Organized into a globally distributed infrastructure over the hierarchical name space, the DNS basically translates human-friendly names into machine-readable network addresses. This way, Internet users and applications heavily rely on the DNS to navigate to the desired services.

The early design of the DNS did not take security and privacy into the central considerations. As the DNS is increasingly turned into a lucrative target for a diversity of attacks, much effort has been devoted into protecting and hardening this critical infrastructure. In comparison with the DNS integrity issues addressed by the on-going transition to the DNS Security Extensions (DNSSEC) [1][2], the DNS privacy concerns and solutions are still not well studied. The DNS requests and responses are never confidential on the Internet. The DNS traffic in the clear is visible to an eavesdropper on the wire. As enablers of DNS communications, authoritative servers and recursive servers are also able to observe, store, and analyze the privacy revealing DNS data. For example, the public DNS recursive servers may readily collect the DNS traffic from a large population of users and analyze their behaviors; the authoritative servers, especially those serving the upper level domains in the DNS tree, can see plenty of full query names in clear text beneath their respective authoritative domains.

In traditional name resolution, a recursive server simply sends a full query name to each authoritative server in a chain of referrals until one of them answers directly. However, the full query name is not always required for any authoritative server to do its name resolution job. For example, the root server doesn't need to know the full query name, say "www.example.com", to do its job; it just needs the TLD (Top Level Domain) "com" , to refer the requester to the authoritative server for that TLD. Since full query names are conventionally and unnecessarily conveyed in every DNS request from a recursive server, it is natural to trim the query names following the principle of minimum disclosure. Thus query name minimization (qname-min) [3] is proposed to minimize the amount of information a recursive server sends to the authoritative server.

In the problem statement of DNS privacy [5] by S. Bortzmeyer, the query name was highlighted as the source of DNS privacy risks. In the document, Bortzmeyer argued that the query names in the DNS traffic gathered by authoritative name servers may be sufficient to violate some privacy expectations. Qname-min [3] was first proposed also by Bortzmeyer in 2014 and agreed by the Internet Engineering Task Force (IETF) in 2016. As of the time of writing, none of the major DNS software vendors has built the qname-min functionality into the production version of resolvers. As an emerging DNS privacy-preserving technique, there are still few studies on the performance measurement and enhancement of qname-min. NXDOMAIN optimization [6], proposed by Vixie et al., was initially intended to improve DNS caching for the conventional name resolution before qname-min was unveiled.

This paper measures the performance of qname-min using the real query names on the Internet. NXDOMAIN optimization is evaluated and found to be an effective technique for improving the performance of qname-min. A DDoS vulnerability of qname-min is identified and its impacts and defenses using NXDOMAIN optimization are evaluated. The broken empty non-terminals (ENTs) are measured and identified as the critical obstacles to the transition to qname-min.

II. Performance

Our basic methodology for studying the performance of qname-min is to probe the authoritative responses to real queries observed by a recursive server. The intermediate query names associated with each full query name were determined as per the qname-min name resolution algorithm. Not only the full query names but also their intermediate query names were used in the active probing. The authoritative responses were identified and analyzed on different qname-min configurations and algorithms.

As a starting point for our measurements, we captured the query logs of a recursive resolver located at the NIST campus

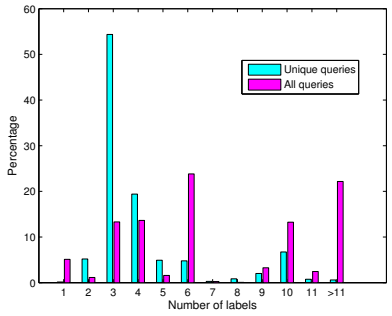

Fig. 1: Distribution of the number of labels for queries.

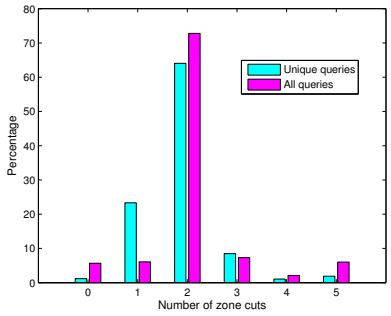

Fig. 2: Distribution of the number of zone cuts for queries.

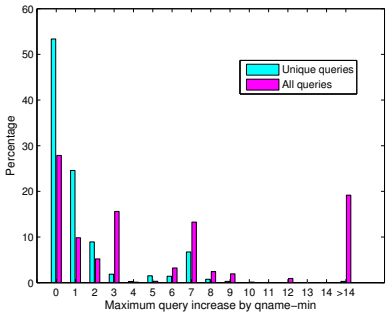

Fig. 3: Maximum query increase by qname-min.

in June 2015. The DNS data contains one-week's queries arriving at the recursive resolver. Using these DNS data, we constructed each query as a pair of the form (query name, query type). This resulted in 42,757 unique queries among a total of 3,908,346 queries.

The performance of qname-min is strongly correlated to the name structure of each individual query name. Generally, a long query name with lots of labels tends to cause high overheads of qname-min. To better understand the measurements, we first present the distribution of the number of labels for queries in Fig. 1. More than 54% of unique query names have three labels, and only a small proportion (no more than 0.6%) of unique query names have more than 11 labels. The average number of labels within an unique query name is 4.18. The distribution of label counts changes greatly if all individual query names are not grouped by the unique query names. For all query names, 6 labels are most concentrated on by 23.8% of the queries, and the query names with more than 11 labels account for 22.2%. The number of labels then rises to 9.56 on average. The measurements show that a small number of long query names attract a large number of queries. So those long query names may degrade the performance of qname-min without aggressive caching in place.

It is not always clear where the administrative zone boundaries in a full query name are set up. As delegations may occur after just one label in some cases or after two (or more) in others, qname-min is not always as simple as just revealing one additional label in the successive queries. However, assuming the recursive server has the knowledge about some zone cuts associated with a query name, qname-min allows the recursive server to start with the closet zone cut and gradually prepend one more label from the full query name as it follows referrals and descend deeper into the domain. Accordingly, another explicit name structure metric that may impact the performance of qname-min is the number of zone cuts within a query name. Generally, for a constant number of labels, more zone cuts increase the chance of starting qname-min probing from a low zone cut point. Thus the number of queries required by qname-min may be saved by the closest zone cut.

To find the zone cut points of a query name, we identify each authoritative response as the following by retrieving and analyzing relevant fields in the DNS response message:

- **NX**: a NXDOMAIN answer. The RCODE is NXDOMAIN.
- **AUTH**: an authoritative answer. The RCODE is NOERROR, the answer section is not empty.
- **REF**: a referral (indicating a zone cut point). The RCODE is NOERROR, the answer section is empty, and the owner name of the NS RRset from the authority section is the query name.
- **NO**: a NODATA answer. The RCODE is NOERROR, the answer section is empty, and the owner name of the NS RRset from the authority section is not the query name.
- **FAIL**: a failure answer. The RCODE is neither NXDOMAIN nor NOERROR, e.g., SERVFAIL.

The REF response indicates a zone cut point above the query name. Together with REF, other types of responses will also enable our measurement and analysis in the following sections.

The number of queries attempted by qname-min is largely dependent of the relevant zone cut points known by the minimizing recursive resolver. In the best efforts to protect the query name's privacy, the minimizing recursive resolver may start at the closest zone cut point pertaining to the full query name, query for just one label more than the closest zone cut point, and progressively performs the label-wise probing until the fully query name is queried for and responded. Like conventional name resolution, the minimizing recursive resolver can cache the zone cut points which may be utilized for qname-min. In Fig. 2, we demonstrate the distribution of the number of zone cuts for captured queries. Two zone cuts consist of a majority, accounting for 64.1% for unique queries and 72.8% for all queries. And the proportion of no more than two zone cuts amounts to 88.6% and 84.6% of unique queries and all queries respectively. That means most queries fall into the SLDs (Second Level Domains) or higher domains. A query name in the bailiwick of a SLD may hide its long label string below that SLD, which has to be probed label-by-label via qname-min despite of the prior knowledge about all relevant zone cuts.

In the worst case of caching, some previously discovered zone cuts may have already expired from cache, or some new zone cuts cannot find their matching counterparts in cache. Without any relevant zone cut information in cache, the minimizing recursive resolver has to start the qname-min from the root zone. Assume that a full query name Q has $\ell(Q)$ labels and $\Re(Q)$ zone cuts. Considering no known zone cuts, the maximum number of queries issued by qname-min is $\ell(Q)$. By contrast, the conventional name resolution only

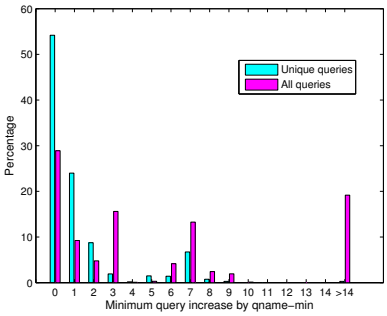

Fig. 4: Minimum query increase by qname-min.

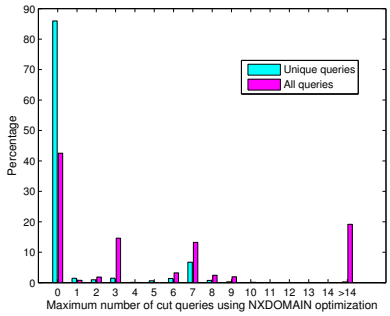

Fig. 5: Maximum number of cut queries using NXDOMAIN optimization.

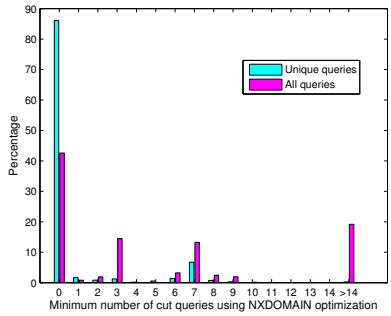

Fig. 6: Minimum number of cut queries using NXDOMAIN optimization.

needs $\Re(Q)+1$ queries. Thus the maximum query increase by qname-min $\Delta_{max}(Q)$ is $\ell(Q)-\Re(Q)-1$. In other words, the maximum overhead of qname-min is determined by the total non-zone-cut-point labels in a query name. Fig. 3 shows the maximum query increase by qname-min. The average of maximum query increase is 1.28 for unique query names while it rises sharply to 6.44 for all query names. The intensive and heavily repeated queries for some long query names contribute much to the qname-min performance drop.

Consider the caching best case where all zone cuts relevant to a full query name are presumably available in cache. For that case, qname-min may skip to the lowest zone cut, which is also the authoritative zone of the query name, as the starting point. So the qname-min probing narrows down to the authoritative zone, using label-by-label progressive queries within the authoritative zone. Assume that a full query name Q has $\ell'(Q)$ labels below the apex of the authoritative zone. Qname-min would need $\ell'(Q)$ queries at minimum to obtain the final answer, but the conventional name resolution always requires one query at minimum regardless of $\ell'(Q)$. So the minimum query increase by qname-min $\Delta_{min}(Q)$ is $\ell'(Q)-1$. Accordingly, a query name with its labels "deep" into its authoritative zone is likely to result in a larger query increase by adopting qname-min. It is obvious that $\ell'(Q)$ is always no more than the total number of non-zone-cut-point labels in a query name, which ensures $\Delta_{max}(Q) \geq \Delta_{min}(Q)$. Fig. 4 shows the minimum query increase by qname-min. Compared with Fig. 3, the distribution of query increase is only slightly changed. The comparative measurements show that the learning, caching, and employing of closest zone cut does not seem to give much improvement to the qname-min's efficiency.

III. NXDOMAIN Optimization

While closest zone cut optimizes the starting point of qname-min, the minimizing recursive server still has to perform progressive probing until the last label. Fortunately, the intermediate name responding specified in the DNS gives the possibilities for further optimizing qname-min performance by truncating the tailing labels. In the DNS, an ENT is referred to as a name that does not have any valid record but has at least one descendant name with valid record(s). For an ENT, the negative response should set the RCODE to NODATA, indicating a valid name, but with no RRsets present. By contrast, all descendant names of an invalid name have no valid records. Thus the negative response to an invalid name should set the RCODE to NXDOMAIN rather than NODATA. That RCODE distinction between ENT (valid intermediate name) and invalid intermediate name enables the early identifying of invalid names in qname-min. More specifically, the minimizing recursive server may interpret the NXDOMAIN response to an intermediate query name as the indicator of the respective invalid full name. Once encountering an intermediate NXDOMAIN response, the minimizing re- cursive server can stop searching down the DNS tree and return a NXDOMAIN response to the full query name. NXDOMAIN optimization can be amplified if NXDOMAIN is not only used on-the-fly but also cached for future responces to any full query names with matching intermediate names. When searching downward in its cache, a recursive server should stop searching if it encounters a cached NXDOMAIN. The response to the triggering query should then be NXDOMAIN. The aggressive use of NXDOMAIN may cut the number of queries as well as the lookup delays for qname-min.

NXDOMAIN optimization eliminates the qname-min probing at the first NXDOMAIN response. So the excessive number of queries required by qname-min will be cut off by NXDOMAIN optimization. We consider the query cut performance with caching effects and without caching effects respectively. With the cached closest zone cut in place, the queries saved by NXDOMAIN optimization are limited to the labels below the first NXDOMAIN label within the authoritative zone. Thus NXDOMAIN optimization gains the minimum benefit for qname-min. As the opposite extreme, without the help of caching, NXDOMAIN optimization will terminate qname-min at the first NXDOMAIN label below the root. Thus the number of queries decreased by NXDOMAIN optimization rises to the maximum. We present the maximum and minimum number of cut queries using NXDOMAIN optimization in Fig. 5 and Fig. 6 respectively. As we can see, there is only a slight difference between the two distributions. In both figures, the query cut effects of NXDOMAIN optimization are much more pronounced for all query names than for unique query names. The average of maximum number of cut queries is 0.81 for unique query names and 6.12 for all query names. For the minimum, the average is almost the same as the maximum: 0.81 for unique query names and 6.12 for all query names. Those results are consistent with our findings in Section 2

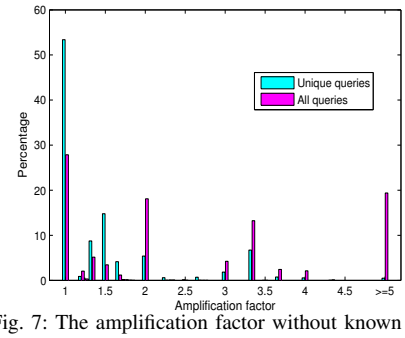

Fig. 7: The amplification factor without known zone cuts.

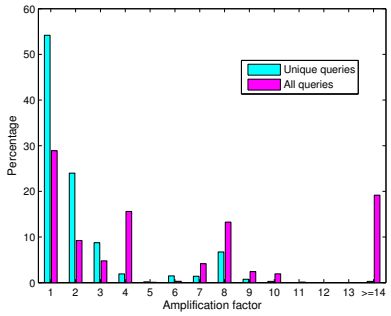

Fig. 8: The amplification factor with known closest zone cut.

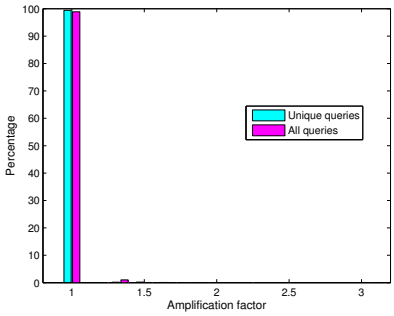

Fig. 9: The amplification factor without known zone cuts by NXDOMAIN optimization.

that the caching of the closest zone cut matters little in terms of qname-min performance. It is notable that some long query names (19.2% in both figures) can be significantly optimized by NXDOMAIN optimization for all query names, leaving more than 14 cut queries.

IV. DDoS Vulnerability

NXDOMAIN optimization is necessary for qname-min not only because of its efficiency but also because of its defense against DDoS attacks. Qname-min increases the attack surface of a recursive server, especially for DDoS attacks. A DDoS attacker can send a flooding stream of DNS queries to the victim recursive server. These queries are subtly constructed for randomly generated but invalid query names with deep labels. Without the NXDOMAIN optimization, qname-min would be likely to perform label-by-label lookups for a number of intermediate names in the resolution for each query name. So a single attacking query will trigger a serial of iterative qname-min DNS lookups by the victim recursive server. Unlike the conventional name resolution, the amplification vulnerability of qname-min cannot be alleviated by caching because the query names may be seldom or never repeated. With the NXDOMAIN optimization, qname-min name resolution can stop early at the most upper NXDOMAIN label of the query name, thereby decreasing its DDoS vulnerability to almost as low as the conventional name resolution.

To make the qname-min based DDoS attacks more devastating, another approach for bypassing the mitigating caching is to register some "cannon" zones dedicated for attacks. The most threatening "cannon" zones should be like: 1) they should reside at the high level of the DNS tree, allowing for more labels below the zone apex (within the length limit to a domain name); 2) they should set small (or even zero) TTLs (Time-To-Lives) of the resource records leveraged for DDoS attacks. Thus attackers, who send flooding queries for the lengthy domains in the "cannon" zones, can trigger tremendous amplified DNS traffic between the targeted recursive server and the authoritative servers. Since authoritative servers of individuals or small businesses are usually hosted by the domain service providers, the domain service providers may also become victims of such DDoS attacks.

A commonly used DNS based DDoS attack is DNS amplification. In a typical DNS amplification attack scenario, the attacker sends relatively small queries that are known to generate much larger responses to the reflecting resolvers. The source IP addresses for the queries are spoofed as the address of the victim so the reflecting resolvers process the recursive queries and deliver the amplified response traffic to the spoofed "origin": the victim. DNS amplification could be exploited to magnify DDoS attack consequences with the origin of the attack concealed from the victim. The amplification capability is basically measured by the amplification factor and is computed as: $\frac{response\ size}{query\ size}$. Unlike DNS amplification attack, the qname-min based DDoS attack can target authoritative servers and recursive servers, which are leveraged by DNS amplification attack as reflectors. Using "deep" query names that are known to produce many times larger number of qname-min probing queries than the conventional name resolution, an attack can create an immense amount of DNS transactions between the (victim) authoritative servers and the (victim) recursive servers. Here we define the amplification factor of qname-min based DDoS attack as: $\frac{the\ number\ of\ \ queries\ in\ qname-min}{the\ number\ of\ queries\ in\ conventional\ name\ resolution}$. The bigger the amplification factor is, the larger the resource consumption is inflicted on the victim.

Assuming all zone cuts are unknown to the recursive server, we present the amplification factor in Fig. 7. The average amplification factors for unique queries and for all queries are 1.47 and 3.15 respectively. Some queries even have a amplification factor as high as 8.19. With known closest zone cut, the amplification factor is illustrated in Fig. 8. The average amplification factors for unique queries and for all queries increase to 2.26 and 7.37 respectively. The largest amplification factor is 25. Both the recursive servers and the respective authoritative servers are expected to handle at least several times larger number of DNS transactions if qname-min is implemented at the recursive servers. In the best efforts of qname-min based DDoS attack, some query names may be carefully selected to have as many as possible labels falling into the target zone. Due to the length limit of a valid DNS name (no more than 253 octets), each of those in-zone labels in the maliciously generated names should be shortened to the minimum, allowing for more in-zone labels. In most cases of qname-min, amplification factors of 30 or more can be easily obtained. The large amplification factors, if employed by DDoS attackers, pose great threats to authoritative servers and/or recursive reservers.

To illustrate the effects of NXDOMAIN optimization on

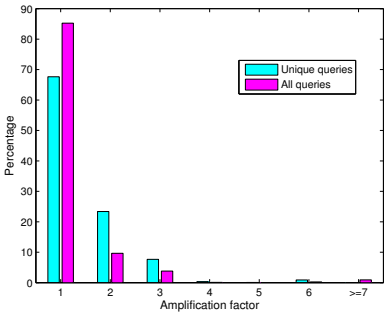

Fig. 10: The amplification factor with known closest zone cut by NXDOMAIN optimization.

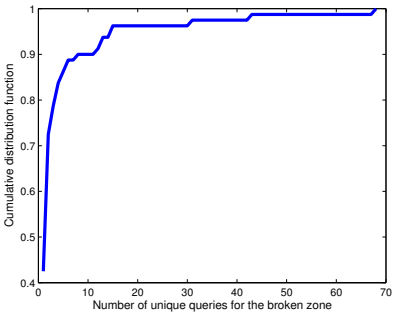

Fig. 11: The identification algorithm of ENT broken query name.

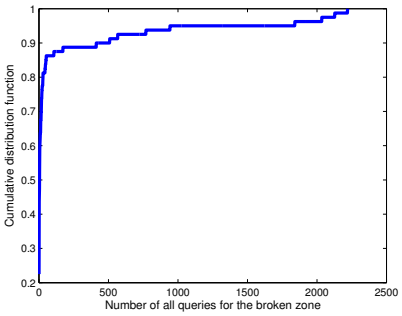

Fig. 12: The identification algorithm of ENT broken zone.

the amplification factor, we present the amplification factor without known zone cuts and with known closest zone cut in Fig. 9 and Fig. 10 respectively. The average amplification factor without known zone cuts is decreased by NXDOMAIN optimization to 1.003 for unique queries and 1.004 for all queries. And for known closest zone cut, NXDOMAIN optimization lowers the average amplification factor to 1.45 for unique queries and 1.24 for all queries. We can see that NXDOMAIN optimization can greatly alleviate the amplification factor to slightly higher than 1. Thus with NXDOMAIN optimization, qname-min has its overheads comparable to conventional name resolution. To get around the caching at recursive servers and trigger outstanding queries from recursive servers, a qname-min based DDoS attack is likely to use (randomly generated) invalid in-zone labels that cause NXDOMAIN responses. Those query names for NXDOMAIN responses make it possible for NXDOMAIN optimization to play its role in defending against qname-min based DDoS attacks.

V. ENT Brokenness

Despite of the significance of NXDOMAIN optimization for qname-min in terms of both performance and security, its use is not always safe assuming authoritative servers would respond properly to ENTs. As defined by the DNS, a query for an ENT should result in the NOERROR RCODE in the response. When a response to an intermediate or non-terminal name has the NXDOMAIN RCODE, it simply means no valid descendant name exists below that name. That is, the standard DNS rules out the possibility that a name with the NXDOMAIN response would find at least one descendant name with RRsets or return a NOERROR response. However, the possibility does emerge if some misbehaving authoritative servers return NXDOMAIN rather than NOERROR responses to ENTs. Since only full query names are typically requested in the conventional name resolution, ENT brokenness has negligible adverse impacts on resolution. However, qname-min with NXDOMAIN optimization may be misled by the erroneous NXDOMAIN response for an intermediate query name into giving up further probing and responding NXDOMAIN as the final response. The consequences of ENT brokenness includes failures/errors of name resolution, or even unavailability and blocking of DNS subtrees.

Assuming all of the name servers for a zone respond the same way, we list a name or zone as broken despite

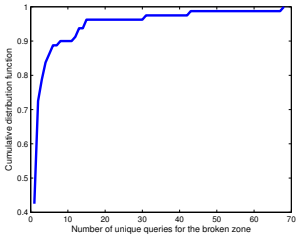

Fig. 13: CDF of the no. of unique queries for the broken zones.

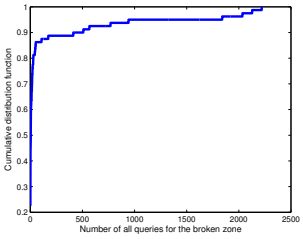

Fig. 14: CDF of the no. of all queries for the broken zones.

the fact that the real DNS responder is one of those name servers. We provide the identification algorithm of ENT broken query name in Fig. 15. The algorithm may start from any intermediate name including the root. However, if the starting name is not the root, only descendant broken ENTs below the starting name can be identified by the algorithm. Like qname-min probing, the algorithm progressively exposes one more label in each successive query. The first phase is dedicated for discovering the first NXDOMAIN returning name. If no NXDOMAIN is found until the last label, the name is identified as good. Otherwise, the successful detecting of the first NXDOMAIN name initiates the second phase: searching for possible NOERROR descendant names. The searching either ends with the first NOERROR descendant name or finally fails to find any NOERROR descendant name until the last label. For the former case, the algorithm returns reporting that the query name is good; for the latter case, the algorithm returns with the detection of a broken query name. The identification algorithm can be integrated into the qname-min algorithm, since the query issuing and response parsing are largely shared between them. Using the identification algorithm, the minimizing recursive server can detect ENT brokenness on-the-fly without any significant extra overhead.

Using the identification algorithm of ENT broken servers, we measured the good responses and the broken responses to queries. The broken responses account for 10.1% of unique query names. This non-trivial proportion illuminates the prevalence of ENT brokenness on the Internet. More surprisingly, the rate of the broken responses increases to 45.1% for all query names. As the explanation to the rise, we find that the ENT broken query names are more likely to have repetitions than the good query names. The root cause in this observation is that clients tend to repeat a query if the presumably erroneous response to the first query does not meet their

Algorithm: FindBrokenDomain(DOMAIN) % Input: a domain to be checked
1: $domain \leftarrow$ " " % Initialize the domain under inspection as the root
2: $res \leftarrow$ NULL % Initialize the response
3: **while** $domain$! = DOMAIN & res ! = NXDOMAIN **do**
4: $domain \leftarrow$ **GetLeftLabel**($domain$, DOMAIN) + "." + $domain$
% Concatenate one more label from the left
5: $res \leftarrow$ **GetResponse**($domain$)
% Query the domain to get the response
6: **if** res = NXDOMAIN **then**
7: **while** $domain$! = DOMAIN & res ! = NOERROR **do**
8: $domain \leftarrow$ **GetLeftLabel**($domain$, DOMAIN) + "." + $domain$
9: $res \leftarrow$ **GetResponse**($domain$)
10: **if** res = NOERROR **then**
% A NOERROR child domain below a NXDOMAIN ancestor is found
11: **Return**("DOMAIN is detected as ENT broken")
12: **else**
13: **Return**("DOMAIN is NOT detected as ENT broken")
14: **else**
% No NXDOMAIN subdomains are found
15: **Return**("DOMAIN is NOT detected as ENT broken")

Fig. 15: The identification algorithm of ENT broken query name.

expectation.

To look further into the sources of ENT brokenness, we seek to figure out what authoritative zones do the broken query names fall into. The identification algorithm of an ENT broken zone is illustrated in Fig. 16. The algorithm can start from any zone cut point, e.g., the TLD, which is initialized as the current zone cut point. The algorithm maintains a set of broken zone cuts, which is initialized as empty. Like the qname-min probing, the algorithm first attempts to find the NXDOMAIN label by progressively prepending labels from the full query name. If no NXDOMAIN is found until the last label, the algorithm outputs the reporting of no broken zone due to the empty set of broken zone cuts. Otherwise, the finding of the first NXDOMAIN switches the searching to the next label. If the algorithm fails to find any NOERROR label until the last label, the set of broken zone cuts is still empty, as the result is a report of no broken zones. Otherwise, the NOERROR label can be further classified into zone cut label and non-zone-cut label. For a zone cut label, the algorithm first adds the current zone cut to the set of broken zone cuts and then updates the current zone cut as the zone cut itself. Using the new zone cut, the algorithm starts over and may potentially add more zone cuts to the set of broken zone cuts. For a non-zone-cut label, the algorithm just adds the current zone cut to the set of broken zone cuts and continues with the searching for zone cut. Note that multiple NOERROR labels within the same current zone cut may result in more than one times of broken zone cut adding, but the set operations themselves can automatically remove all duplicate members and ensure the uniqueness. In all cases, the algorithm continues until the last label and outputs the set of broken zones as it terminates. The identification algorithm can also be embedded into the qname-min algorithm, since the progressive probing results of qname-min can be simply parsed and interpreted by the identification algorithm.

Using the identification algorithm of ENT broken zones, we measured the number of queries for each broken zone. The cumulative distribution function of the number of unique

Algorithm: FindBrokenZones(DOMAIN, $zone$) % Input: a domain to be checked, and a zone to start with ($zone$ should be " " for a complete run)
1: $domain \leftarrow zone$ % Initialize the domain under inspection
2: $BrokenZones \leftarrow \varnothing$ % Initialize the broken zone set as empty
3: $res \leftarrow$ NULL % Initialize the response
4: **while** $domain$! = DOMAIN & res ! = NXDOMAIN **do**
5: $domain \leftarrow$ **GetLeftLabel**($domain$, DOMAIN) + "." + $domain$
% Concatenate one more label from the left
6: $res \leftarrow$ **GetResponse**($domain$)
% Query the domain to get the response
7: **if** res indicates a zone cut **then**
8: $zone \leftarrow domain$ % Find a new zone to inspect
9: **if** res = NXDOMAIN **then**
10: **while** $domain$! = DOMAIN & res ! = NOERROR **do**
11: $domain \leftarrow$ **GetLeftLabel**($domain$, DOMAIN) + "." + $domain$
12: $res \leftarrow$ **GetResponse**($domain$)
13: **if** res = NOERROR **then**
% A NOERROR child domain below a NXDOMAIN ancestor is found
14: $BrokenZones \leftarrow BrokenZones + zone$
15: **if** res indicates a zone cut **then**
16: $BrokenZones \leftarrow BrokenZones +$ **FindBrokenZones**(DOMAIN, domain) % Find a new zone to inspect
17: **Return**($BrokenZones$)

Fig. 16: The identification algorithm of ENT broken query name.

queries and all queries for the broken zone are illustrated in Fig. 13 and Fig. 14 respectively (excluding the heaviest hitter). We can see that the distribution is highly skewed towards a small number of hot zones. The findings imply that while the few broken zones can be relatively easy to discover because of their intensive query traffic, a majority of broken zones are much more difficult to detect.

VI. Conclusion

In this work, we provide a measurement study on qname-min. We find that the privacy earning by qname-min potentially comes at the cost of name resolution performance. Since the query increase by qname-min is largely companioned by NXDOMAIN responses, NXDOMAIN optimization is found to be effective in cutting the number of queries in qname-min. A DDoS vulnerability exploiting large qname-min amplification factors is discovered to pose serious threats to authoritative servers and/or recursive servers. And NXDOMAIN optimization plays a defensive role against such vulnerability. We find the widespread presence of broken ENTs and identify them as the sources of failures/errors of qname-min name resolution. The study highlights at least two requirements for qname-min implementations: 1) NXDOMAIN optimization should be adopted; 2) good robustness against ENT brokenness should be provided.

References

[1] Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: Resource Records for the DNS Security Extensions. IETF RFC 4034 (2005).
[2] Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: Protocol Modifications for the DNS Security Extensions. IETF RFC 4035 (2005).
[3] Bortzmeyer, S.: DNS Query Name Minimisation to Improve Privacy. IETF RFC 7816 (2016).
[4] Lewis, E.: The Role of Wildcards in the Domain Name System. IETF RFC 4592 (2006).
[5] Bortzmeyer, S.: DNS Privacy Considerations. IETF RFC 7626 (2015).
[6] Vixie, P., Joffe, R., Neves, F.: Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness. IETF Draft in Progress (2010).