On Aggressive Negative Caching in DNS Resolvers

Casey Deccio
Computer Science Department
Brigham Young University
Provo, UT 84602
Email: casey@byu.edu

Benjamin Tessem
Computer Science Department
Brigham Young University
Provo, UT 84602
Email: btessem@byu.edu

***Abstract*—Caching has been a fundamental feature of the Domain Name System (DNS) since its inception. Resolvers cache the response to a query – whether that response was a name-to-resource mapping or a code indicating that no mapping exists. Recent additions to the DNS include a behavior known as aggressive negative caching, wherein resolvers simply infer that no mapping exists, based on previous responses, thus saving a query to authoritative servers. In this paper we perform the first known study of aggressive negative caching in the wild. We issue experimental queries to resolvers associated with 2,500 world-wide RIPE Atlas probe to detect the behavior in real resolvers. We observe aggressive negative caching in roughly half of the resolvers we analyzed. We also perform an analysis of several open source DNS resolver implementations, which show both that behaviors differ between implementations and that implementations are often consistent with their own documentation.**

I. Introduction

The Domain Name System (DNS) is the set of protocols, servers, and software that provide the critical name-to-address mapping for the Internet. Since its inception over 40 years ago, the DNS has undergone many changes, each of which has affected the Internet ecosystem in some way. Many changes have been about improving security and privacy; others have been about increasing efficiency. One change that was driven by both is aggressive negative caching.

Caching DNS responses has been an integral part of the DNS since its design. A DNS resolver saves records that it has retrieved from authoritative servers, until they expire. This includes negative responses—i.e., those that indicate that a name has no mapping. Aggressive negative caching is a similar idea, but the negative responses are *inferred* from previously cached information, rather than explicitly learned through DNS queries. For example, a DNS resolver might learn through its queries to other DNS servers that names alphabetically between `a.example.com` and `d.example.com` do not have mappings; when queried later for `b.example.com`, it might elect to issue an `NXDOMAIN` (name not found) error directly rather than explicitly querying to find the answer. This is aggressive negative caching.

Aggressive negative caching has been shown to significantly reduce the quantity of queries for nonexistent domain names between DNS resolver and DNS authoritative server [1]. This is especially true at the root and top-level domain levels because of the large fraction of queries destined for those servers that are for nonexistent domain names [2]. This reduction in queries between resolvers and authoritative servers is certainly an improvement in efficiency. However, it also enhances organizational privacy because authoritative servers (third parties) observe fewer queries issued by resolvers (organizations).

While this behavior is generally seen as an improvement, it is not without consequence. For example, many researchers in the Internet community rely on datasets such as Day in the Life of the Internet (DITL), which is an annual collection of DNS queries at the DNS root servers, over a 48-hour period [3]. The absence of a large number of queries for nonexistent domains could prove to be a significant gap in analyses of DITL data, depending on their purposes. Knowing the prevalence of DNS resolvers that exhibit this behavior is important for better understanding the DNS and Internet ecosystem, including the value of datasets like DITL.

In this paper we measure the presence of aggressive negative caching behaviors in various locations around the Internet. Using a collection of globally distributed RIPE Atlas probes, we issue queries against a carefully designed DNS infrastructure and analyze the results. The major contributions of this paper are the following:

- A novel methodology for detecting aggressive negative caching;
- An Internet-wide measurement of aggressive negative caching behavior; and
- A study of aggressive negative caching behavior in various open-source DNS resolver implementations.

We find that roughly half of the resolvers we analyze exhibit behaviors consistent with aggressive negative caching, including some public DNS resolver services. We also observe a variety of implementation behaviors, which both distinguish one implementation from another and put an implementation at odds with its documented features; in nearly every case, the observed behaviors for an implementation do not align with its documentation.

II. Background

The DNS [4], [5] involves three main components of infrastructure: stub resolver, recursive resolver, and authoritative server. The stub resolver exists as a software library on an end-user system. It issues queries to a recursive resolver, which either answers from its cache or finds the answer by querying one or more authoritative servers. The authoritative servers hold databases mapping domain names (e.g., `foo.com`) and types to resources: type `A` for a IPv4 address, type `AAAA` for an IPv6 address, type `MX` for mail exchange records, etc. This database is referred to as a DNS zone.

All DNS queries consist of a domain name and a type. DNS responses likewise include the queried domain name and type, as well as a response code that is typically one of the following: `NOERROR` (name exists); `NXDOMAIN` (name does not exist); `SERVFAIL` or `FORMERR` (server encountered an error while processing the query). Finally, if there is a resource corresponding to the queried name and type, then it is returned in the answer section of the response. An empty answer section constitutes a negative response. Both answers and negative responses can be cached by a recursive resolver. The maximum time that an answer or negative response can be cached is specified in a time-to-live (TTL) value returned in the response.

The DNS Security Extensions (DNSSEC) [6], [7], [8] provide origin authentication to the DNS. With DNSSEC, responses include cryptographic information with which a resolver can validate their validity. A more detailed analysis of DNSSEC can be found in other works. However, a component of DNSSEC that is especially relevant to this work is authenticated denial of existence. When a negative response is issued, the response contains a set of `NSEC` (next secure) records that indicate the name(s) that *exist*, on either side of the queried name (based on a specified canonical ordering), which does *not* exist. These records constitute the nonexistence proof for the queried name. `NSEC3` records are similar but instead of providing a plain-text proof, they contain the hashes of existing names; these hashes are on either side of the hash of the queried name (based on a specified canonical ordering), which does not exist.

In the case of both `NSEC` and `NSEC3`, the nonexistence proofs that accompany negative responses typically provide enough evidence for the nonexistence of other names to be inferred as well. For example, if a server returns the proof that there is nothing between `a.foo.com` and `d.foo.com`, in response to a query for `b.foo.com`, it can infer that `c.foo.com` is also nonexistent. This inference was initially prohibited by DNSSEC specification [8]. However, that restriction was lifted in an updated specification [9].

We note that some authoritative DNS servers implement minimal `NSEC` or `NSEC3` proofs, referred to as "black lies" and "white lies", respectively [10], [11]. In these cases, rather than returning `NSEC` or `NSEC3` records that correspond to domain names that actually exist, `NSEC` or `NSEC3` records are synthesized on demand to minimally "cover" the queried domain (`NSEC`) or its hash (`NSEC3`). In both of these cases, the proofs returned from authoritative servers resolver are so minimal that they cannot be used for inferring nonexistence of any other domain names.

III. Previous Work

Both DNSSEC signing and DNSSEC validation have been studied since the early days of DNSSEC deployment. Osterweil, et al., were among the pioneers in this area, measuring the adoption of DNSSEC deployment during its early years [12]. Later DNSSEC measurement studies focused on the quality and analysis of deployment, with an attempt to understand not only the adoption rate but also the reasons for its slow uptake [13], [14], [15]. Measurement studies associated with DNSSEC validation have also been carried out [16], [17]. In this paper, we measure the pervasiveness of DNSSEC validation, but mostly in conjunction with our study of resolver and aggressive negative caching.

DNS negative caching was the subject of a 2019 study, in which the resolvers of more than 7,000 RIPE Atlas probes were analyzed to see if they cached a negative response [18]. A more recent (2024) study included not only traditional negative caching (`NXDOMAIN` and `NODATA`), but also different server behaviors, such as other response codes (e.g., `FORMERR`) and other response contents (e.g., no `SOA` record in the authority section) [19]. There is some overlap between these previous studies and this paper. Negative caching is studied in this paper, but it is only incidental to aggressive negative caching, which is the primary focus of this paper.

Very little work has been done in studying `NSEC`, `NSEC3`, and aggressive negative caching. While not a study of aggressive negative caching, Demke, et al., studied the deployment of `NSEC` and `NSEC3` in DNSSEC-signed zones, including black lies and white lies—from which no aggressive caching could take place [20]. The only study of aggressive negative caching of which we are aware was presented as a "lightning talk" by Huston, et al., in 2019 [21].

One of the byproducts of aggressive negative caching is additional organizational privacy; only a fraction of the queries resulting in negative responses issued to an organization's resolver are also seen by the authoritative servers. Query name (QNAME) minimization is another DNS extension that also results in reduced data shared with authoritative servers. While surveys of aggressive negative caching have not been found in academic forums, several studies of QNAME minimization have been carried out in recent years [22], [23], [24].

IV. Measurement Methodology

Our basic methodology for determining whether or not a given resolver implements aggressive negative caching is the following:

1) Create a DNSSEC-signed zone with a single, large gap between names within the zone.
2) Issue queries to the DNS resolver for unique, nonexistent query names within the single large gap in the zone.

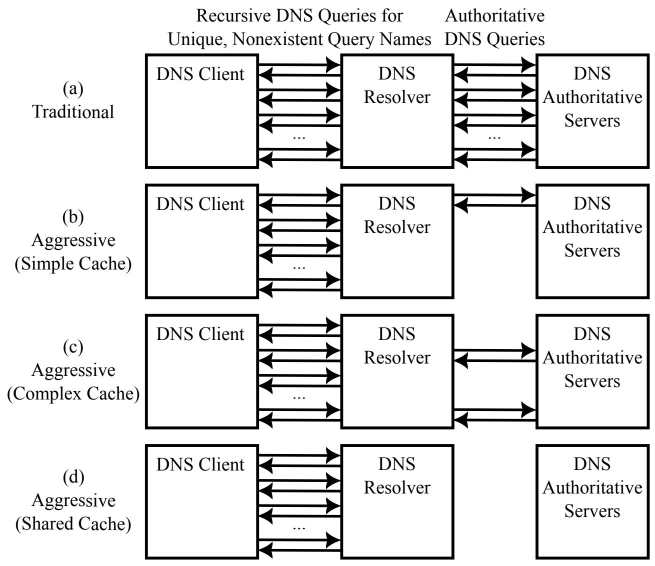

Recursive DNS Queries for Unique, Nonexistent Query Names
Authoritative DNS Queries
(a) Traditional
(b) Aggressive (Simple Cache)
(c) Aggressive (Complex Cache)
(d) Aggressive (Shared Cache)
DNS Client
DNS Resolver
DNS Authoritative Servers
...

Fig. 1: Different resolver behaviors associated with negative caching. Scenario (a) illustrates traditional negative caching. Scenarios (b) through (d) illustrate aggressive negative caching with a simple cache, complex cache, and shared cache, respectively.

3) Correlate authoritative DNS queries—or lack thereof—with the recursive queries issued.

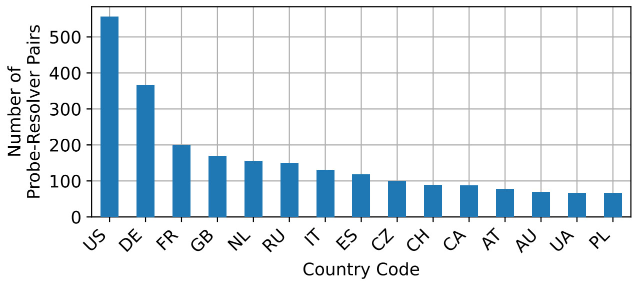
In the case that a resolver implements aggressive negative caching, we expect the authoritative servers to see queries for fewer query names than the number of `NXDOMAIN` responses than the client receives from the recursive resolver. This is illustrated in Figure 1, with scenario (a) representing traditional caching and scenarios (b) through (d) representing different variants of aggressive negative caching. These scenarios are further explained in Section V.

In the rest of this section we describe the set of resolvers that we selected for analysis as well as the DNS zones and queries used to detect negative caching behavior.

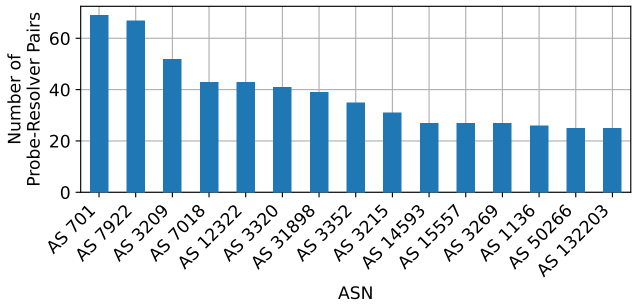
A. Resolver Selection

We used RIPE Atlas probes to analyze DNS resolvers deployed around the world. We selected 2,500 probes using the "Random by Area" option, and specifying the "WW (World-Wide)" area. We further specified that each probe should issue queries to each of its configured DNS resolvers. While there are several datasets from which we might have chosen our resolvers, we used RIPE Atlas because it provides DNS query access to resolvers in a diversity of geographic and network locations, including both public and non-public DNS resolvers.

In all, 3,940 unique probe-resolver pairs were identified for analysis. With those probe-resolver pairs, 132 countries and 1,264 autonomous system numbers (ASNs) were represented. The mean and median probe-resolver pairs per country were 38 and 35, respectively. The mean and median probe-resolver pairs per ASN were 160 and 118, respectively. The top 15 represented countries and ASNs are shown in Figure 2. The



(a)



(b)

Fig. 2: Distribution of probe-resolver pairs in (a) the top 15 countries and (b) the top 15 ASNs.

United States is the most represented, followed by Germany and France. The most represented ASNs correspond to Internet service providers Verizon (AS 701), Comcast (AS 7922), Vodaphone (AS 3209), and AT&T (AS 7018).

While there are 3,940 unique probe-resolver pairs, the number of actual resolvers is fewer because there is some resolver redundancy. In some cases a single probe might be configured with multiple IP addresses that actually reference the same recursive resolver. For example, they might use `192.0.2.1` and `2001:db8::1`, which could correspond to the IPv4 and IPv6 address of a single resolver. It could also be that a given resolver is used by multiple probes. In these cases, if aggressive negative caching is employed, it is possible that we observe no queries at our authoritative servers for a given probe-resolver pair, i.e., because the shared resolver has the nonexistence proof cached before the "first" query from that pair. In any case, this study is to provide some basis for measuring aggressive negative caching, even if the dataset has limitations.

Of the 3,940 probe-resolver pairs, about one third (1,292 or 33%) are associated with three well-known public DNS resolver services by their primary IPv4 or IPv6 addresses: Cloudflare (374 or 9%); Google (640 or 16%); and Quad9 (278 or 7%). The infrastructure for these resolver services is more complex, such that a resolver that employs aggressive negative caching might generate a set of queries representing less than 100% of the query names, instead of just a single query. In this case, sending multiple queries increases the chances of hitting an intermediate cache with the nonexistence

Zone	DNSSEC Algorithm	NSEC Type	DNSSEC Status	Per-Query False Positive Rate
NSEC-S	8	`NSEC`	secure	0.0
NSEC-I	8	`NSEC`	insecure	0.0
NSEC3-S	8	`NSEC3`	secure	9.3×10^{-10}
NSEC3-I	8	`NSEC3`	insecure	9.3×10^{-10}
BROKEN	8	N/A	bogus	N/A

TABLE I: Summary of characteristics of DNS zones used for testing aggressive negative caching.

proof. Ultimately, however, the results is based on the number of backend caches. There might be false negatives if no two queries are forwarded to the same backend cache. However, the chance for false positives is much smaller because it would effectively be caused by only by authoritative queries that were not logged due to packet loss.

B. Experimental DNS Zones

In this section we describe the DNS zones used for issuing queries to detect aggressive negative caching. Section IV implies a single DNS zone for which queries would be issued. However, we actually create multiple zones, which can collectively help us expand our understanding beyond just the base question of whether or not a resolver implements aggressive negative caching. For example, we would like to know whether a resolver supports aggressive negative caching with `NSEC`, `NSEC3`, or both, and whether a resolver carries out aggressive negative caching without DNSSEC validation or without a valid chain of trust (i.e., contrary to specification).

We now describe the purpose and configuration of the five DNS zones we created. A summary is found in Table I.

NSEC-S. Our baseline zone, NSEC-S, is DNSSEC-signed using `NSEC`. Also, there is a complete chain of trust, such that the negative responses can be fully authenticated by a DNSSEC-validating resolver, for a status of "secure" [8]. The NSEC-S zone includes only two names: the zone name itself (`foo.com`); and a domain name that closely follows the zone name in canonical ordering (`\007.foo.com`, where `\007` is a byte with value 7) [7]. A query for effectively *any* proper subdomain of the zone name (e.g., `a.foo.com`, `b.foo.com`) results in an `NXDOMAIN` response. The `NSEC` record returned in response to any such query indicates that there are no names between `\007.foo.com` and `foo.com`. That record alone is proof enough for any resolver that uses aggressive negative caching to infer an `NXDOMAIN` response to further queries for proper subdomains of `foo.com`, without re-querying the NSEC-S authoritative servers. We make two notes about the use of `\007` as the first label of the name. First, this is a nod to Cloudflare's black lies for minimally-sized nonexistence proofs [11], even though this is used for the exact opposite purpose—to create the largest gap in names possible. Second, we originally used `\000`, to create the maximum possible gap. However, we found that one resolver implementation—Knot Resolver—did not perform aggressive negative caching with the `\000` label, possibly because treating it differently "improves cache effectivity with DNSSEC black lies" [25]—even though our configuration is the very opposite of black lies, with the `\000` being associated with the owner name of the `NSEC` record, instead of the `Next Domain Name` field.

NSEC-I. The second DNS zone, NSEC-I, is configured identically to NSEC-S, except that there is no full DNSSEC chain of trust. This means that responses cannot be fully authenticated by a DNSSEC-validating resolver, which is a requirement for aggressive negative caching [9]. Queries to this zone are for determining whether or not implementers are following this requirement, in accordance with specification.

NSEC3-S. The third DNS zone, NSEC3-S, is DNSSEC-signed and has a complete chain of trust, like NSEC-I. However, it uses `NSEC3` instead of `NSEC`. In order to get the large gap in names, as we did for NSEC-S (and NSEC-I), we used millions of loop iterations to generate a domain name whose hash was relatively close to the hash of the zone name. In this case, the hash does not *immediately* follow the hash of the zone name, as is the case with NSEC-S; with an off-the-shelf authoritative DNS server implementation like the one we used in our experiment, such would have been basically impossible. However, we quantifiably minimized the gap between the two hashes using the following logic. The hash associated with an `NSEC3` record is 1,024 bits, so the hash space is 2^{1024}. The "difference" between the hash of the generated name and the hash of the zone name covers all but 9.3×10^{-10} of the hash space. This means that, on average, roughly 1 in a billion queries are likely to fall outside of that range and thus not be "covered" by that `NSEC3` record. For our experiment, that figure becomes our false positive rate for an individual query. However, as we describe in Section IV, multiple queries are issued to test a given resolver, and the false positive rate further decreases when multiple queries are issued. The NSEC3-S zone is used to see if a given resolver behaves differently with `NSEC3` vs. `NSEC` records, as far as aggressive negative caching is concerned.

NSEC3-I. The fourth DNS zone, NSEC3-I, is configured identically to NSEC3-S, with the exception that there is no full DNSSEC chain of trust. This is the equivalent of NSEC-I, but for `NSEC3`.

BROKEN. The final DNS zone, BROKEN, is DNSSEC-signed. It also uses `NSEC`, but the use of `NSEC` and `NSEC3` is irrelevant for this zone; its purpose is not to infer the use of aggressive negative caching but to infer whether or not a resolver is performing DNSSEC validation. The chain of trust for this zone is deliberately broken, such that any responses from a DNSSEC-validating resolver should have response code `SERVFAIL` because of a validation status of "bogus" [8].

All zones are signed with DNSSEC algorithm 8 (RSA-SHA1-NSEC3), as are the zones in their chain of trust, including the top-level domain (TLD) and the root zone. We used algorithm 8 because it provides the best likelihood that it is supported by resolvers for validation, which is a minimum requirement for aggressive negative caching. Resolvers might not support all currently deployed algorithms, but any resolver that wants to anchor with the root zone keys must support algorithm 8. The negative response TTL for all zones is four hours.

C. DNS Queries

To test each probe-resolver pair for behaviors indicative of aggressive negative caching, we issue it a total of 37 queries, consisting of the following:

- **Aggressive Negative Cache Queries.** a sequence of 8 queries, each for a unique, nonexistent query name within each of the NSEC-S, NSEC-I, NSEC3-S, and NSEC3-I zones; and
- **DNSSEC Queries.** 5 queries, each for a unique, nonexistent query name within the BROKEN zone.

Queries within each zone were spaced 1-minute apart, which is the smallest interval allowed by RIPE Atlas. The sequences of queries to each of the zones were issued in parallel; that is, the queries for NSEC-S, NSEC-I, NSEC3-S, NSEC3-I, and BROKEN were all issued during the same time period.

The query name for each query was a proper subdomain of one of our experimental zones, i.e., by adding a single label to the left of the zone name. To ensure uniqueness of the query names across probes and resolvers, this left-most label follows convention `$t-$p-experimentid`, where `$t` is the current timestamp, `$p` is the identifier of the probe issuing the query, and `experimentid` is a unique identifier associated with this set of measurements. A resolver without aggressive negative caching would be expected to forward every query to the authoritative servers, which are under our control (i.e., the traditional negative caching behavior (a) in Figure 1). In contrast, a resolver that practices aggressive negative caching would be expected to forward only the first query to the authoritative servers, after which the nonexistence of subsequent queries would be inferred from the `NSEC` or `NSEC3` (i.e., one of the aggressive negative caching behaviors (b) through (d) in Figure 1). In either case, we would see evidence of this in our logs.

We note that adding only a single label to the zone name to create the query name is important because recent protocol clarifications specify that the nonexistence of subdomains (e.g., `c.b.a.foo.com`) can be inferred from the nonexistence of their superdomains (e.g., `a.foo.com`), even without the use of aggressive negative caching [26]. Adding only a single label helps us avoid false positives.

V. Measurement Results

We now analyze the results of our measurement study. Because the probes did not receive or report a response for every recursive query issued, we only considered the measurement result for a given probe-resolver pair if the probe received at 8 `NXDOMAIN` responses from the resolver. Of all probe-resolver pairs, 3,617 (92%) qualified in the NSEC-S experiment and 2,998 (76%) qualified in the NSEC3-S experiment. The total qualifying resolvers in at least one of the experiments was 3,653, corresponding to the 93% of the total. This is summarized along with other results in Table II.

Description	Unique Probe-Resolver Pairs NSEC-S	NSEC3-S	All
Total	3,934	3,924	3,940
8 `NXD` Responses for either NSEC-S or NSEC3-S	3,617 (92%)	2,998 (76%)	3,653 (93%)
Traditional Neg. Caching	1,827 (51%)	2,894 (97%)	3,227 (88%)
Aggressive Neg. Caching	1,790 (49%)	104 (3.5%)	1,793 (49%)
Simple cache	489 (27%)	5 (4.8%)	491 (27%)
Complex cache	616 (34%)	96 (92%)	702 (39%)
Shared Cache	685 (38%)	3 (2.9%)	685 (38%)
No DNSSEC Validation	0 (0%)	0 (0%)	0 (0%)
No chain of trust	0 (0%)	0 (0%)	0 (0%)
8 `NXD` Responses for both NSEC-S and NSEC-3	2,962 (75%)		
Aggressive Neg. Caching	1,468 (50%)	101 (3.4%)	1,468 (50%)
NSEC-S only	1,367 (93%)	N/A	N/A
NSEC3-S only	N/A	0	N/A

TABLE II: Results of empirical analysis of resolvers associated with RIPE Atlas probes. Percentages shown are in relation to the bold heading most closely above and less indented than the percentage. Because the "All" column is a union of the results in the "NSEC-S" and "NSEC3-S" columns, the sum of the subcomponents of the "All" column can exceed 100%.

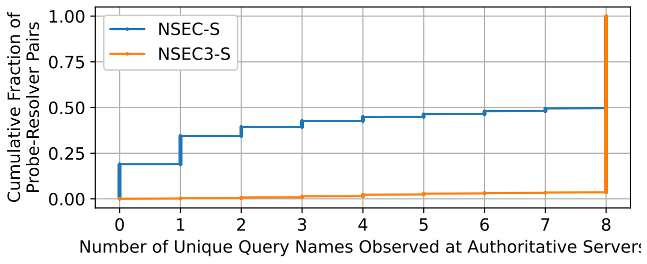


Fig. 3: Cumulative distribution of the number of unique query names in queries observed at authoritative servers, for queries issued to NSEC-S and NSEC-I zones.

We analyzed the number of unique query names observed in queries at the authoritative servers, on a per-DNS-zone basis. A plot showing the cumulative distribution of unique names observed for NSEC-S and NSEC3-S is shown in Figure 3. The plot and table both show that about 49% of probe-resolver pairs exhibited behavior consistent with aggressive negative caching with NSEC-S—i.e., the query names for fewer than 8 queries were observed at our authoritative servers. Fewer than 5% of probe-resolver pairs exhibited behavior matching the pattern of aggressive negative caching with NSEC3-S. We investigate this disparity further between NSEC-S and NSEC3-S further in Section VI.

We consider three different specific patterns that we attribute to aggressive negative caching, illustrated in Figure 1, as scenarios (b), (c), and (d), respectively:

- **Simple cache.** Only the first of the queries resulting in

Description	Probe-Resolver Pairs
Total	3,940
5 SERVFAIL **or** NXDOMAIN **Responses**	3,741 (95%)
All SERVFAIL **Responses**	2,731 (73%)
All NXDOMAIN **Responses**	959 (26%)
Mix of SERVFAIL **and** NXDOMAIN **Responses**	51 (1.4%)

TABLE III: Results of empirical DNSSEC validation testing of resolvers associated with RIPE Atlas probes. Percentages shown are in relation to the bold heading most closely above and less indented than the percentage.

NXDOMAIN responses were observed at the authoritative servers.
- **Complex cache.** At least one of the queries resulting in NXDOMAIN responses were observed at the authoritative servers. This excludes the behavior of the "simple cache", i.e., that only authoritative query associated with the first NXDOMAIN response was observed.
- **Shared cache.** No queries were observed at the authoritative servers.

Of the probe-resolver pairs exhibiting aggressive negative caching with NSEC-S, just over one quarter (27%) fit the pattern of simple cache—i.e., only a single query name was observed at the authoritative servers. The other 73% were resolvers with complex caches and shared caches. For NSEC3-S, 95% of probe-resolver pairs performing aggressive negative caching were complex and shared caches.

We also looked for some behaviors that are inconsistent with the specification for aggressive negative caching. We found no examples of aggressive negative caching with NSEC-I or NSEC3-I—i.e., with no chain of trust. Neither did we find examples of resolvers that employed aggressive negative caching without DNSSEC validation, which would have been manifested by NXDOMAIN (instead of SERVFAIL) responses to queries in the BROKEN zone.

We observed that 73% of probe-resolver pairs appear to support DNSSEC validation, as evidenced by consistent SERVFAIL responses after a query to our BROKEN zone. Another 26% of probe-resolver pairs returned consistent NXDOMAIN responses, an indicator that they do not support DNSSEC validation. The remaining 1.4% returned a mixture of SERVFAIL and NXDOMAIN, the inconsistency of which might again be the result of multiple backends that are configured differently as far as validation is concerned. This is summarized in Table III.

VI. Resolver Behavior Characterization

Having studied behaviors of Internet resolvers, we now examine the behaviors of individual implementations, for a comparison. Several open-source implementations have implemented aggressive negative caching since 2018, including BIND (since version 9.12.0), Unbound (since version 1.7.0), Knot Resolver (since version 2.0.0), and PowerDNS Recursor (since version 4.5.1). Table IV contains a summary of major DNS resolver implementations and their adoption of aggressive negative caching, according to their own documentation.

Notably, the addition of this feature has been far from all-or-none. The history of aggressive negative caching in each of the open source implementations, according to their own documentation, is as follows. BIND originally implemented aggressive negative caching for both NSEC and NSEC3 in 2018 and made it the default configuration [27]. However, in 2019, it was disabled by default due to it having "a significant performance impact" [31]. In 2021, aggressive negative caching was restored as the default behavior in BIND, but only support for NSEC was included [34]. Unbound implemented aggressive negative caching in 2018 but did not make it the default behavior until 2022 [28], [32]. Finally, Knot Resolver first implemented aggressive negative caching for only NSEC and later added functionality for NSEC3 [29], [33]. Only PowerDNS Recursor has supported aggressive negative caching for both NSEC and NSEC3 continuously, by default, since it was initially introduced.

Using the documentation as a starting point, we test each of the open-source implementations for aggressive negative caching behavior. We also test the behavior of three major public DNS providers: Cloudflare, Google, and Quad9. The rest of this section describes our methodology and findings.

A. Methodology

For each DNS resolver software implementation, we selected the latest minor version in each major version, beginning with versions that implement aggressive negative caching as default behavior and ending with the most recent release:
- BIND 9.17.22, 9.18.35, 9.19.24, 9.20.7, 9.21.6
- Unbound 1.17.1, 1.18.0, 1.19.3, 1.20.0, 1.21.1, 1.22.0
- Knot Resolver 3.2.1, 4.0.0, 4.1.0, 4.2.2, 4.3.0, 5.0.1, 5.1.3, 5.2.1, 5.3.2, 5.4.4, 5.5.3, 5.6.0, 5.7.4
- PowerDNS Recursor 4.5.12, 4.6.6, 4.7.6, 4.8.4, 4.9.9, 5.0.9, 5.1.3, 5.2.0

For Unbound, Knot Resolver, and PowerDNS Recursor, we deployed each software version in its own Docker container. For BIND, we compiled each version from source. Each resolver was configured to perform DNSSEC validation and aggressive negative caching. We also issued tests to several major public DNS resolver services, each from two different vantage points, one near Salt Lake City and one near San Francisco, both in the United States: Cloudflare (1.1.1.1), Google (8.8.8.8), and Quad9 (9.9.9.9).

For each software version and each public DNS vendor we issued the following queries, with reference to Section IV-B:

Aggressive Negative Cache Queries. 100 queries (200 for public DNS services), each for a unique query name, within each of the NSEC-S, NSEC-I, NSEC3-S, and NSEC3-I zones. This amounts to 400 queries (800 for public DNS services).

DNSSEC Queries. 5 queries to the BROKEN zone. If the resolver is configured to perform DNSSEC validation, then these will result in SERVFAIL responses.

	BIND		Unbound		Knot Resolver		PowerDNS Recursor	
config. option	`synth-from-dnssec`		`aggressive-nsec`		(None)		`aggressive-nsec-cache-size`	
	Version (year)	NSEC/3	**Version (year)**	NSEC/3	**Version (year)**	NSEC/3	**Version (year)**	NSEC/3
documented features	9.12.0 (2018 [27])	● ●	1.7.0 (2018 [28])	◐ ◐	2.0.0 (2018 [29])	● ○	4.5.1 (2021 [30])	● ●
	9.15.6 (2019 [31])	◐ ◐	1.15.0 (2022 [32])	● ●	2.4.0 (2018 [33])	● ●		
	9.17.21 (2021 [34])	● ○						

TABLE IV: A summary of aggressive negative caching implementation in open-source DNS resolver software, according to their documentation. Support for both `NSEC` and/or `NSEC3` is shown as follows:
○ not implemented; ◐ implemented but not default behavior; ● implemented and enabled by default.

Negative Cache Queries. 100 queries (200 for public DNS services) for each of five unique query names, within each of the NSEC-S, NSEC-I, NSEC3-S, and NSEC3-I zones. This amounts to 2000 queries (4000 for public DNS services). While basic negative caching has been a feature of the DNS from the very beginning, we issued these tests to check for behavioral differences across the different software versions, particularly when `NSEC` or `NSEC3` are used in the DNS zones. The expectation is that each of the unique query names results in only a single query at our authoritative servers—or 0 queries, if its nonexistence could be inferred without even a single query, with the help of aggressive negative caching. For more complex caches, like those employed in public DNS services, we might expect to observe more than a single query at our authoritative servers because the 200 queries were distributed across multiple backend caches.

B. Results

In our analysis of the open-source resolver implementations, we saw only three behaviors associated with aggressive negative caching: only a *single* unique query name is observed in authoritative queries (i.e., scenario (b) in Figure 1), *all* query names are observed (i.e., scenario (a) in Figure 1), and about 60% of query names are observed. The first two cases are straight-forward examples of aggressive negative caching and traditional negative caching; the latter is the aggressive negative caching behavior specific to Knot Resolver, and we explain it subsequently.

With the public DNS services, we saw the following patterns: all query names observed or between 1 and 10% of query names observed. We attribute the range (1–10%) to the cache complexity of the public DNS services; multiple queries to a given front-end will not necessarily be routed to the same backend cache. Thus, even with aggressive negative caching, a second query to the front-end does not always result in induced nonexistence. However, the percentage for services providing aggressive negative caching is clearly distinguishable.

The full results of our experimentation are shown in Table V. Every row represents the observed behavior of a given set of resolver implementations, specified by the "Software" and "Version" columns. The rows are unique, with one exception: the rows representing the observed behaviors of PowerDNS Recursor 4.9.9 through 5.20 are identical to those of BIND 9.17.22 through 9.18.35.

One of the biggest observations from our experimentation is that none of the software implementations exhibit the negative caching behavior described in its documentation. The documentation for every implementation we tested indicates support for aggressive negative caching with `NSEC`. Based on our analysis, such support was detected in all implementations, with the exception of BIND 9.19.24 and later—even though previous versions of BIND supported it. Similarly, we anticipated aggressive negative caching behavior with `NSEC3` for all Unbound, Knot Resolver, and PowerDNS Recursor versions, based on their documentation. Yet none of the Unbound versions we tested appeared to support this feature. PowerDNS supported aggressive negative caching with `NSEC3` in versions through 4.8.4, but it was not observed in 4.9.9 and beyond.

Knot Resolver supports aggressive negative caching with `NSEC3` in all versions, but with some caveats. For roughly 60% of query names issued to a DNS zone signed with `NSEC3`, Knot Resolver made no attempt at aggressive negative caching—or even normal negative caching. For the remaining (approximately) 40% of query names, aggressive negative caching and normal negative caching worked as expected for the DNS zones signed with `NSEC3`. Whether negative caching and aggressive negative caching is carried out by Knot Resolver for `NSEC3` is consistent for a given query name. That is, a given name was found to always support negative caching (aggressive or not) or never support it. This leads us to believe that the decision is based on the hash of a query name.

In our testing of public DNS resolver services, we observed that Google and Quad9 both supported aggressive negative caching with `NSEC` (only), while Cloudflare had no apparent support for aggressive negative caching with either `NSEC` or `NSEC3`. These behaviors were consistent across the two vantage points from which we queried.

We now consider the number of queries issued to authoritative servers in conjunction with a negative response. We calculated the average number of queries observed at our authoritative servers per query name sent to the server. For example, if 600 queries were observed at our authoritative servers for 100 domain names, then the average would be 6. With the exception of Unbound versions 1.17.1 through 1.18.0, all resolver implementations, including public DNS services, issue only a single query to authoritative servers when the zone is signed with `NSEC`, whether NSEC-S or NSEC-I. With `NSEC3` zones, the average number of authoritative queries associated with a negative response to the resolver ranges

			Negative Caching				Aggressive Negative Caching				Avg. Authoritative Queries Per Query Name			
		DNSSEC	NSEC		NSEC3		NSEC		NSEC3		NSEC		NSEC3	
Software / Vendor	Version / Year	Validation	S	I	S	I	S	I	S	I	S	I	S	I
BIND	9.17.22 (2022) – 9.18.35 (2025)	●	●	●	●	●	●	○	○	○	1	1	2	2
	9.19.24 (2024) – 9.21.6 (2025)	●	●	●	●	●	○	○	○	○	1	1	2	2
Unbound	1.17.1 (2023) – 1.18.0 (2023)	●	●	●	●	●	●	○	○	○	2	1	3	3
	1.19.3 (2024) – 1.22.0 (2024)	●	●	●	●	●	●	○	○	○	1	1	6	6
Knot Resolver	3.2.1 (2019) – 5.7.4 (2024)	●	●	●	◐	●	●	○	◐	○	1	1	4	2
PowerDNS	4.5.12 (2022) – 4.8.4 (2023)	●	●	●	●	●	●	○	●	○	1	1	2	2
Recursor	4.9.9 (2024) – 5.2.0 (2025)	●	●	●	●	●	●	○	○	○	1	1	2	2
Cloudflare	March 24, 2025	●	●	●	●	●	○	○	○	○	1	1	6	6
Google	March 24, 2025	●	●	●	●	●	●	○	○	○	1	1	3	3
Quad9	March 24, 2025	●	●	●	●	●	●	○	○	○	1	1	3	3

TABLE V: Results of empirical testing of DNS resolver implementations: ○ no support; ◐ partial support; ● support.

from 2 to 6, for both NSEC3-S and NSEC3-I. In the case of Knot Resolver (only), the average number of queries observed differs between NSEC3-S and NSEC3-I, with 4 and 2 queries, respectively.

Several behaviors were consistent across all resolver implementations and public DNS services. All resolvers were DNSSEC-validating. No implementation carried out aggressive negative caching in either of the insecure zones, NSEC-I and NSEC3-I. Finally, with the exception of Knot Resolver, all performed regular negative caching as expected.

We note that the general lack of support for aggressive negative caching in the presence of `NSEC3` across public DNS providers and open source implementations is consistent with the findings in Section V, wherein the 3.5% aggressive negative caching rate with `NSEC3` is significantly lower than the 49% rate found with `NSEC`.

VII. Discussion

Based on our analysis of Internet resolvers using RIPE Atlas probes, approximately half of deployed resolvers appear to employ aggressive negative caching for `NSEC`. Additionally, all of the open source implementations that we analyzed claim to exhibit this behavior by default, with all but one actually matching that claim. Thus, for the operator of a DNS zone signed with `NSEC`, any query analysis should take into account that the authoritative queries resulting in negative responses are quite possibly just a subset of the recursive queries—not just in terms of rate (a consequence of normal negative caching), but also in terms of query names observed in authoritative queries. The exact fraction of query names observed by a given resolver is a function of three things: the pattern of queries to the resolver, the negative cache TTL, and the zone contents themselves. A resolver that is queried more frequently for a diversity of nonexistent names in the zone is more likely to exhibit aggressive negative caching behavior than one that is queried less frequently for names in the zone or for a less diverse set of query names. This is simply a matter of probabilities; with more nonexistent query names to choose from, there is a greater statistical likelihood that one nonexistent name falls into the coverage associated with the `NSEC` records of another. A higher negative cache TTL for the zone is more likely to result in higher instance of aggressive negative caching; as long as `NSEC` records remain in cache, they are possible matches for nonexistent query names. Finally, if the names within a zone are somewhat evenly distributed, with relatively small gaps in between, nonexistent names queried are statistically less likely to fall into those gaps and thus be inferred by aggressive negative caching; whereas, for a zone with large gaps, the probability is higher. The measurement of these variables—query frequency and diversity, negative cache TTL, and zone name distribution—are not addressed in the current study but are the subject of future work.

For zones signed with `NSEC3`, the factors contributing to the amount of aggressive negative caching are the nearly the same, but as shown in Section V, only about 3% of resolvers perform aggressive negative caching with `NSEC3`. Thus, for operators of zones using `NSEC3`, the set of nonexistent names in recursive queries is more likely to be reflected in authoritative query logs.

The extent to which authoritative server operators care about the accuracy of their authoritative server logs, in terms of queries for nonexistent names, might vary from operator to operator. For example, the DITL dataset [3] is used by a community of Internet researchers, and a more comprehensive dataset results in more representative analysis. If the operator of a zone signed with `NSEC` (or `NSEC3` wishes to reduce the effects of aggressive negative caching, there are some factors under its control. It could arbitrarily increase the names in the zone to create smaller nonexistence gaps or decrease the negative cache TTL. Alternatively, it could deploy a white lies [10] or black lies [11] approach, to effectively eliminate aggressive negative caching altogether, albeit with some additional deployment cost.

VIII. Ethical Considerations

With any measurement experiment, it is important to behave responsibly and ethically, and to minimize any potential harm, for which the perceived benefits outweigh the potential harms. With this experiment, the primary ethical concern was the burden on third-party infrastructure. In this case, that refers to the RIPE Atlas infrastructure, the DNS resolvers with which the associated probes are configured, and the public DNS services. For the DNS queries performed on RIPE Atlas (Section IV) our maximum send rate for probe resolvers was only five DNS queries per minute, per probe-resolver pair. Each probe would have seen a multiple of this rate, based on the number of resolvers with which it was configured. A given resolver might also see a multiple of this rate if it is used by multiple probes. For the resolver implementation analysis in Section VI, we issued queries in parallel, 20 or more seconds apart, a small burden for a public DNS service. All these things considered, we assess that the burden placed on both RIPE Atlas and the involved resolvers to be negligible.

IX. Conclusion

In this paper we have studied the prevalence of aggressive negative caching in DNS resolver implementations and deployments. Such behavior is exhibited by roughly half of the resolvers we analyzed with RIPE Atlas. Additionally, we have found that different open-source DNS resolver implementations exhibit mostly unique behavior, with regard to aggressive negative caching, traditional negative caching, and the number of queries generated from the resolver in connection with a negative response. We also observed that documented behavior of these implementations differs from observed behavior in almost every case.

While the study presented herein is somewhat limited in terms of what can be generally applied, it provides a foundation for future work to better understand this question and area. We hope that this and future work will help understand usage models, as well as the implications of deploying new protocols in a security- and privacy-centered Internet.

Acknowledgments

We gratefully acknowledge the contributions of Tyler Karren for his help on early versions of this work, as well as RIPE NCC for the use of the RIPE Atlas platform. We also acknowledge the TMA 2025 reviewers and our shepherd for their helpful guidance in preparing our paper for publication.

References

[1] W. Kumari, "draft-ietf-dnsop-nsec-aggressiveuse," 2016. [Online]. Available: https://www.ietf.org/proceedings/96/slides/slides-96-dnsop-2.pdf

[2] G. Huston, "The Root of the DNS," Feb 2017. [Online]. Available: https://www.potaroo.net/ispcol/2017-02/roots.html

[3] Domain Name System Operations, Analysis, and Research Center, "Day In The Life of the Internet (DITL)," 2025. [Online]. Available: https://www.dns-oarc.net/oarc/data/ditl

[4] P. Mockapetris, "RFC 1034: DOMAIN NAMES - CONCEPTS AND FACILITIES," November 1987.

[5] ——, "RFC 1035: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION," November 1987.

[6] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "RFC 4033: DNS security introduction and requirements," March 2005.

[7] ——, "RFC 4034: Resource records for the DNS security extensions," March 2005.

[8] ——, "RFC 4035: Protocol modifications for the DNS security extensions," March 2005.

[9] K. Fujiwara, A. Kato, and W. Kumari, "RFC 8198: Aggressive Use of DNSSEC-Validated Cache," July 2017.

[10] D. Kaminsky, "Phreebird." [Online]. Available: https://dankaminsky.com/phreebird/

[11] D. Grant, "Economical with the truth: Making DNSSEC answers cheap." [Online]. Available: https://blog.cloudflare.com/black-lies/

[12] E. Osterweil, M. Ryan, D. Massey, and L. Zhang, "Quantifying the operational status of the dnssec deployment," in *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 231–242. [Online]. Available: https://doi.org/10.1145/1452520.1452548

[13] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra, "Quantifying and improving dnssec availability," in *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, 2011, pp. 1–7.

[14] T. Chung, R. van Rijswijk-Deij, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "Understanding the role of registrars in dnssec deployment," in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 369–383. [Online]. Available: https://doi.org/10.1145/3131365.3131373

[15] M. Wander, "Measurement survey of server-side dnssec adoption," in *2017 Network Traffic Measurement and Analysis Conference (TMA)*, 2017, pp. 1–9.

[16] Y. Yu, D. Wessels, M. Larson, and L. Zhang, "Check-repeat: A new method of measuring dnssec validating resolvers," in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 3147–3152.

[17] K. Fukuda, S. Sato, and T. Mitamura, "A technique for counting dnssec validators," in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 80–84.

[18] L. Shafir, Y. Afek, A. Bremler-Barr, N. Peleg, and M. Sabag, "DNS negative caching in the wild," in *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*, ser. SIGCOMM Posters and Demos '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 143–145. [Online]. Available: https://doi.org/10.1145/3342280.3342338

[19] M. Zeng, Y. Zhu, B. Li, Y. Zhang, Q. Liu, and B. Fang, "Failed yet stored: A first look at DNS negative caching," in *2024 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2024, pp. 1–10.

[20] J. Demke and C. Deccio, "On dnssec negative responses, lies, and zone size detection," in *Passive and Active Measurement*, D. Choffnes and M. Barcellos, Eds. Cham: Springer International Publishing, 2019, pp. 231–243.

[21] G. Huston, "NSEC Caching Revisited," 2019. [Online]. Available: https://indico.dns-oarc.net/event/32/contributions/717/attachments/713/1206/2019-10-31-oarc-nsec-caching.pdf

[22] W. de Vries, Q. Scheitle, M. Müller, W. Toorop, R. Dolmans, and R. van Rijswijk-Deij, "A First Look at QNAME Minimization in the Domain Name System," in *Passive and Active Measurement. PAM 2019.*, D. Choffnes and M. Barcellos, Eds. Cham: Springer International Publishing, 2019, pp. 147–160.

[23] J. Magnusson, M. Müller, A. Brunstrom, and T. Pulls, "A second look at dns qname minimization," in *Passive and Active Measurement: 24th International Conference, PAM 2023, Virtual Event, March 21–23, 2023, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2023, p. 496–521. [Online]. Available: https://doi.org/10.1007/978-3-031-28486-1_21

[24] A. Hilton, C. Deccio, and J. Davis, "Fourteen years in the life: A root Server's perspective on DNS resolver security," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 3171–3186. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/hilton

[25] CZ.NIC, "Knot resolver 4.1.0," July 2019. [Online]. Available: https://knot-resolver.readthedocs.io/en/stable/NEWS.html#knot-resolver-4-1-0-2019-07-10

[26] S. Bortzmeyer and S. Huque, "RFC 8020: NXDOMAIN: There Really Is Nothing Underneath," Nov 2016.

[27] I. S. Consortium, "Bind 9.12.0 released," January 2018. [Online]. Available: https://ftp.isc.org/isc/bind9/9.12.0/CHANGES

[28] N. Labs, "Unbound 1.7.0," March 2018. [Online]. Available: https://nlnetlabs.nl/projects/unbound/download/#unbound-1-7-0

[29] CZ.NIC, "Knot resolver 2.0.0," January 2018. [Online]. Available: https://knot-resolver.readthedocs.io/en/stable/NEWS.html#knot-resolver-2-0-0-2018-01-31

[30] PowerDNS, "Powerdns recursor 4.5.1," May 2021. [Online]. Available: https://doc.powerdns.com/recursor/changelog/4.5.html#change-4.5.1

[31] I. S. Consortium, "Bind 9.15.6 released," November 2019. [Online]. Available: https://ftp.isc.org/isc/bind9/9.15.6/CHANGES

[32] N. Labs, "Unbound 1.15.0," February 2022. [Online]. Available: https://nlnetlabs.nl/projects/unbound/download/#unbound-1-15-0

[33] CZ.NIC, "Knot resolver 2.4.0," July 2018. [Online]. Available: https://knot-resolver.readthedocs.io/en/stable/NEWS.html#knot-resolver-2-4-0-2018-07-03

[34] I. S. Consortium, “Bind 9.17.21 released,” December 2021. [Online]. Available: https://ftp.isc.org/isc/bind9/9.17.21/CHANGES