CAMP: Compositional Amplification Attacks against DNS

Huayi Duan[1], Marco Bearzi[1], Jodok Vieli[1],
David Basin[1], Adrian Perrig[1], Si Liu[1], and Bernhard Tellenbach[2]

[1]ETH Zurich
[2]Armasuisse

Abstract

While DNS is often exploited by reflective DoS attacks, it can also be weaponized as a powerful amplifier to overload itself, as evidenced by a stream of recently discovered application-layer amplification attacks. Given the importance of DNS, the question arises of what the fundamental traits are for such attacks. To answer this question, we perform a systematic investigation by establishing a taxonomy of amplification primitives intrinsic to DNS and a framework to analyze their composability. This approach leads to the discovery of a large family of compositional amplification (Camp) vulnerabilities, which can produce multiplicative effects with message amplification factors of hundreds to thousands. Our measurements with popular DNS implementations and open resolvers indicate the ubiquity and severity of Camp vulnerabilities and the serious threats they pose to the Internet's crucial naming infrastructure.

1 Introduction

As a public UDP service with an amplification effect, DNS has been frequently *abused* to launch reflective DoS attacks. Yet, as a foundational component of the Internet, DNS itself is also an inviting *target* for DoS attacks: a DNS outage can disrupt numerous online services. For instance, the infamous Mirai Dyn attack in 2016 rendered over one hundred thousand popular domains unavailable to tens of millions of users [22].

DNS-based reflection attacks leverage simple forms of *transport-layer* amplification: response messages are larger than request messages [38, 37], and an oversized response will trigger TCP fallback and hence a few extra messages [15]. Such generic amplification is also present in many other UDP-based protocols [34].

In contrast, emerging DoS attacks against DNS servers reveal more sophisticated forms of *application-layer* amplification: a single client request triggers a large number of resolver queries, as a result of allowable DNS features such as cascading referrals or query rewrites [5, 26, 29, 11]. Such attacks require the setup of DNS zones with specially crafted records on the target or auxiliary nameservers, which is easily achievable with today's popular and accessible DNS hosting services. Previous amplification vulnerabilities have been discovered mostly by the ad-hoc analysis on some particular aspect of DNS dynamics (see the discussion in Section 3). The sheer complexity of DNS protocols suggest the existence of many more such vulnerabilities yet to be unveiled.

The above motivates a systematic investigation of application-layer amplification vulnerabilities inherent to DNS. We start by establishing a taxonomy of *amplification primitives*, which prolong name resolution with excessive queries of distinct forms such as concurrent and chaining ns queries. This approach allows us to both characterize all existing vulnerabilities of this kind and uncover subtle new attack vectors.

Next, we thoroughly examine the *composability* of amplification primitives. This leads to the discovery of a large family of *compositional amplification* (Camp) vulnerabilities, where each derivative name of a primitive generates another primitive and so on. They can produce much higher *multiplicative* amplification effects than individual primitives. We exhaustively establish 16 valid two-dimensional (2D) compositions based on our taxonomy; they are *regular* in that all primitives in the second dimension are of the same type and size. The results from our composability analysis also enable the construction of exponentially many *irregular* and *multi-dimensional* compositions.

We demonstrate the perils of Camp with extensive measurement of three major DNS implementations (BIND, Unbound, and PowerDNS) and over 50 open resolvers. All of them are vulnerable to multiple Camp variants: 2D compositions can already achieve a message amplification factors (MAF) of several hundreds for many resolvers; the MAF of 3D compositions can ramp up to several thousands. We further illustrate the

high efficiency of distributed CAMP attacks by simulating them against our own authoritative nameservers. These results are highly concerning given the variety of CAMP, the prevalence of accessible DNS hosting services, and the existence of millions of exploitable open resolvers.

Mitigation. Mitigating CAMP attacks is non-trivial. Most compositions do not violate the standard DNS protocols specified in RFCs; in fact, some of them result from the protocol specifications' ambiguities, e.g., when query name minimization should be applied. Only one particular feature—assigning aliases to NS names—exploited by a few compositions is prohibited by the RFCs yet is still allowed by many practical implementations. Hence, to eliminate CAMP at the protocol level, a general consensus must be reached within the broad DNS community.

An essential mitigation mechanism is to restrict the resolver queries generated for the resolution of each client request. Such limits (e.g., `max-recursion-depth`) have been implemented by resolver software in response to previous amplification vulnerabilities. However, CAMP vulnerabilities stand out in that they can bypass individual limits and achieve an MAF equal to the multiplication of these limits. This necessitates the design of holistic query limiting algorithms that account for the subtle interactions between DNS features.

We propose and discuss a list of mitigation options in Section 6, ranging from protocol-level patches and configuration options that cap the MAFs of CAMP vulnerabilities, to system-wide rate limiting and anomaly detection that can counter distributed CAMP attacks.

Disclosure. We have reported our findings to the software vendors of the three DNS implementations used in our measurements. They have confirmed our results and are now working with us to implement and assess the mitigation options (more details in Section 6). A dedicated backchannel within the DNS-OARC Mattermost forum was established to facilitate the discussion. In particular, BIND and Unbound have fixed their query limiting mechanisms and will release the patches around mid August. We are planning to invite more DNS-OARC members into the discussion.

Moreover, we are working with a national cyber security authority to coordinate the responsible disclosure to broader communities. This includes (1) informing the network operators of critical infrastructure and DNS service providers within our country of residency, (2) issuing security advisories to the authority's international partners, and (3) disclosing the vulnerabilities to the general public. In the meantime, the authority will assist in the assignment of CVE and/or CWE numbers.

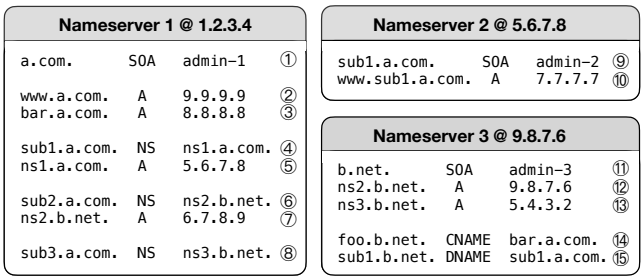


Figure 1: A DNS configuration for three zones containing the main types of RRs. Each zone is hosted by a separate nameserver. Below are examples for different response types. The notation `Q`, T → `IP` : C, n means that a query for name `Q` of type T to nameserver at address `IP` elicits a response with code C and an RR indexed by n.

- `www.a.com.`, A → `1.2.3.4` : NOERROR, 2
- `qqq.a.com.`, A → `1.2.3.4` : NXDOMAIN, 1
- `www.sub1.a.com.`, A → `1.2.3.4` : NOERROR, 4 5
- `www.sub2.a.com.`, A → `5.6.7.8` : REFUSED, –
- `foo.b.net.`, A → `9.8.7.6` : NOERROR, 14
- `www.sub1.b.net.`, A → `9.8.7.6` : NOERROR, 15

2 DNS Preliminaries

DNS is a distributed database over a tree-structured namespace, where a name like `www.example.com.` consists of dot-delimited labels. The namespace is partitioned into administrative units (subtrees) called *zones*. Each zone can independently manage names within its authority and delegate parts of its authority to create subzones. By convention, the zones right below the root are called top-level domains (TLDs) such as `com` and `net`, which typically serve delegation-only data to create second-level domains (SLDs) like `example.com`.

A zone's data is organized as *Resource Records* (RRs) that are stored in a zone file. RRs map names to data of different types: network addresses (type A/AAAA for IPv4/IPv6), servers for zone delegation (type NS), other (canonical) names (type CNAME/DNAME), etc. Records of the same name and type but with different data are grouped as an RRset. Each zone has one Start of Authority (SOA) record that defines the zone's meta information.

An *authoritative nameserver* (hereafter, just nameserver) hosts zone files and responds to name lookup queries with matching RRsets. In practice, DNS clients do not directly send requests to nameservers but instead rely on a *recursive resolver* (hereafter, just resolver) that interacts with nameservers to find authoritative answers for clients. Caching RRs at resolvers significantly reduces the amount of queries sent to nameservers, especially those associated with the root and TLDs, and thus makes DNS scalable to serve the entire Internet.

The interaction between resolvers and nameservers is

Table 1: Summary of amplification primitives in each distinct form and variants of their construction.

Atomic Form	Amplification Primitive	Zone Setup			Reported first in
		Records	# Zone	# Server	
Fan-out	Concurrent referral	Glueless/Out-of-bailiwick NS RRset	1	1	[26, 5]
	Failover referral	Same as above, but the NS names own RRs	1	1	this work
Chaining	Referral chain	Chaining glueless/out-of-bailiwick NS RRs	1	1	[26]
			≥2	≥2	[29]
	Rewrite chain	Chaining CNAME/DNAME RRs	2	2	[11]
			≥1	1	[25]
Self-probing	QNAME minimization	One RR owned by a deep name	1	1	[8, 14]
	Dense delegation	One NS RR per label of a deep name	≥2	≥2	this work

the most intricate part of the name resolution process. Upon receiving a client request, if a resolver fails to find an answer directly from its cache, it will ask the most relevant nameserver it knows, and possibly many others, until it decides on a final response to the client. This process is often idealized in textbooks as an *iterative resolution*, where the resolver contacts nameservers on the delegation path of the query name (QNAME) in a top-down manner. However, the process is substantially more complex in reality due to the recursive nature of DNS: *the resolution of one name can lead to and depend on the resolution of other names.*

For each query a resolver sends to a nameserver, the response falls within one of the four possibilities below.

- **Definite answer.** This concludes the query either positively with a matching RRset, or negatively with an NXDOMAIN (non-existing domain) status code indicating that the QNAME (and any node below it in the namespace [9]) does not exist.
- **Referral.** This contains an NS RRset referring the resolver to other nameservers responsible for the QNAME. If the response contains no IP addresses for these servers or such data is deemed untrustworthy (explained in Section 3.1), the resolver must resolve their names as well.
- **Rewrite.** This means that the QNAME is an alias mapping to no meaningful data other than the canonical name in the returned CNAME/DNAME RR. The resolver should continue to resolve the name.
- **Failure.** An error code such as SERVFAIL or REFUSED indicates that the responding nameserver cannot process the query for some reason (technical issues, security policies, etc.). Error handling at the resolver varies across implementations, e.g., retrying another available nameserver.

A definite answer terminates the resolution of a query, whereas a referral, rewrite, or failure response can prolong the process by triggering new queries. Figure 1 give examples for all these cases.

3 Taxonomy of DNS Amplification

The hierarchical structure and recursive nature of DNS makes name resolution an inherently complicated process. We focus on analyzing application-layer DNS amplification vulnerabilities, where the resolution of a single name generates excessive queries. Since these queries can interleave with each other and also recursively trigger further queries, enumerating all possible ways of amplification presents an obvious challenge.

We tackle this by identifying distinct forms of amplification based on the relations between a QNAME (which we term *base name*) and the subsequent names (which we term *derivatives*) queried for its resolution. Depending on whether a base name and its derivatives belong to the same path in the namespace, and whether the derivatives can be independently queried, we can define three distinct forms that cover all possibilities: *self-probing*, *fan-out*, and *chaining*. We then enumerate DNS features that instantiate each form, establishing a taxonomy of *amplification primitives*, as summarized in Table 1 and elaborated in this section. The taxonomy serves as the basis for us to analyze how these primitives interact with each other (Section 4).

This paper does not consider transport-layer amplification (e.g., TCP fallback) and implementation bugs (e.g., [6, 2]) as well as temporal DoS attacks [4, 10]. They are orthogonal to, and can be potentially combined with, application-layer amplification vulnerabilities.

3.1 Fan-out

A fan-out effect occurs when a name's resolution triggers *parallelizable* queries for other names. That is, the triggered queries can be issued independently, and possibly at different times. This requires that the base name and the derivatives are not suffixes of each other.

Among all RR types, NS, CNAME, and DNAME are the only types that map a name to another name while influencing the DNS resolution process. By definition, each

name should have just one CNAME or DNAME RR, the existence of which prohibits any other RR of the same name. This restriction does not apply to NS: DNS requires each zone to be hosted by at least two nameservers for availability, and there are 13 *logical* root nameservers (each backed by an anycast constellation of servers). Therefore, fan-out is only possible with NS RRs.

Note that a referral response does not always lead to fan-out. An NS RR in a response can be accompanied by a *glue record*—an A/AAAA RR that maps the nameserver's name (NS name for short) to its address. If an NS name is under the same zone delegated by the corresponding NS RR (e.g., 4 in Figure 1), a glue record should be provided to avoid circular dependency; in this case, the NS name is regarded *in-bailiwick* (viz., within the answering zone's authority) and the associated glue is considered trustworthy [36]. For security reasons, a resolver should discard the glue record of an otherwise *out-of-bailiwick* NS name (e.g., 6 in Figure 1) and initiate separate resolution to obtain the nameserver's IP address. It is also possible that an NS RR comes as *glueless* without any glue record at all (e.g., 8 in Figure 1); a resolver should then separately resolve these NS name as well.

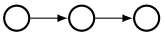
3.1.1 Concurrent Referral

Fan-out caused by a large glueless or out-of-bailiwick NS RRset was initially reported by Maury [26]. Later, Afek et al. [5] conducted a detailed analysis of this vulnerability, showing that for performance reasons resolvers attempt to proactively resolve all NS names in a referral response at once. They proposed a mitigation approach, MaxFetch(k), that limits the number of such concurrent NS queries to k for each client request. MaxFetch(k) has since been adopted by major resolver implementations.

3.1.2 Failover Referral

In addition to the above-mentioned concurrent version, fan-out can also happen sequentially. If a resolver fails to obtain the IP address of a seemingly resolvable nameserver (e.g., when the NS name owns a CNAME RR) or any useful answer from a resolved nameserver (e.g., a SERVFAIL response or a lame delegation [7]), it will try another nameserver. This natural failover mechanism (stated in the initial RFC1034 [27]) is overlooked in previous studies [26, 5]. It is activated after the initial concurrent queries for NS names and thus can bypass the limits enforced by MaxFetch(k). It turns out that resolver implementations are more generous to sequential failover queries than performance-oriented concurrent queries, making the former a more powerful amplifier.

3.2 Chaining



Chaining occurs when a name's resolution results in *recursive* queries for different names that are not suffixes of each other. These queries are generated in order. While fan-out expands a resolver's queries in breadth, chaining goes in depth. A chaining primitive should be constructed by RRs of a single type. A chain with more than one RR type is not considered as a primitive but rather a composite form of amplification.

3.2.1 Referral Chain

A referral chain consists of a sequence of glueless or out-of-bailiwick NS RRs. A resolver receives them one at a time and resolves the NS names sequentially. This is first exploited by the indefinite DNS (iDNS) attack [26], where a referral chain is constructed at a single zone. The chaining NS RRs can also reside in multiple zones hosted by different servers. A loop is formed if two names on a chain depend on each other. This is exploited by the TsuNAME attack [29], where a resolver keeps querying a set of nameservers hosting zones with cyclical delegations. However, most resolver implementations can detect such circular dependencies and terminate the resolution process [29]. Hereafter, we consider all chains (including referral and rewrite) to be loop-free.

3.2.2 Rewrite Chain

A rewrite chain consists of consecutive CNAME or DNAME RRs. These two RR types differ in that the former rewrites a name in its entirety, whereas the latter replaces a name's suffix (i.e., rewriting an entire zone). Since they are equivalent in terms of the construction of rewrite chains, hereafter we focus on CNAME only.

The Unchained attack [11] exploits a CNAME chain where names alternate between two zones hosted by separate nameservers. When matching a CNAME RR for a query, a nameserver will try to look up the rewritten name again in the zones it hosts and pack all related CNAME RRs into the response [27]. This allows a resolver to rewrite the names locally without sending extra queries. Splitting a rewrite chain across two servers necessitates the individual resolution of all names on the chain.

However, we observe that many resolver implementations actually discard all but the first RR on a rewrite chain carried in a response to prevent cache poisoning attacks. Therefore, it is possible to exploit a rewrite chain for amplification by employing just one nameserver that hosts all the CNAME RRs either in a single zone or multiple zones. This doubles the amplification effect that Unchained induces on a target nameserver.

3.3 Self-Probing

In contrast to fan-out and chaining, where different names are resolved, self-probing occurs when a resolver keeps sending queries for the *same* name or its suffixes. This is indeed a normal behavior during iterative resolution: for example, a resolver sends the same query for `www.a.com.` iteratively to the root, TLD, and SLD nameservers, and receives in-bailiwick referrals for `com.` and `a.com.`, and the final authoritative answer for `www.a.com.`, respectively. There are two ways to yield an amplification effect in the self-probing form.

3.3.1 Query Name Minimization

Conventionally, a resolver always sends the full QNAME of a client request when querying nameservers. This enables nameservers, especially those at the top of the DNS hierarchy, to collect massive amount of DNS traffic and track user activities. Query Name Minimization (QMIN) addresses this privacy concern by having a resolver ask nameservers with the minimal labels of a QNAME. For example, a QMIN-enabled resolver will send only `com` to the root server and `a.com.` to the TLD server, preventing them from learning more information than necessary.

However, QMIN introduces a new DoS risk: for a deep QNAME with many labels, a resolver may repeatedly ask the same nameserver with an increasing number of labels until the final RR is found, as illustrated in Figure 10a. While this vulnerability is generally recognized [8, 14, 39], and placing limits on the maximum number of QMIN iterations is mandated [8], we find that QMIN is a more subtle amplifier than previously assumed, as it can interact with the query rewrite and referral mechanisms.

3.3.2 Dense Delegation

A deep name can stimulate amplification even in the absence of QMIN. The idea is to set up zone delegation at each level of the name and trigger a prolonged iterative resolution process, as illustrated in Figure 10b. These zones should be hosted by different nameservers, because a server hosting multiple zones always uses the zone matching the longest suffix of a QNAME to find an answer, precluding the extra queries we expect. We refer to this primitive as dense delegation (DDLG) for the artificial and unusual delegation behavior.

It might appear that DDLG can be exploited to target a resolver only, since the query load is distributed to nameservers. Yet, we demonstrate that it can be combined with other primitives to concentrate queries towards a single nameserver as well (Section 4.2). Unlike a long referral chain that is sometimes regarded as abnormal, DDLG follows the legitimate iterative resolution mechanism without incurring recursion, and resolvers generally place no direct limits on it. This vulnerability is therefore more difficult to fix than others.

Table 2: Summary of regular multiplicative composability of 5 amplification primitives. F.O. is short for fan-out, R.C. for referral chain, W.C. for rewrite chain, Q.M. for query minimization, and D.D. for dense delegation. The primary primitive has x derivatives and each instance of the secondary primitive has y derivatives. The last three columns display the minimum number of nameservers required for setup and a single client request's amplification effect on a resolver and a nameserver. The rows are compressed to save space.

Prim.	Secd.	# Name Servers	# Queries from resolver	# Queries to focal
F.O.	D.D.	$1+y$	$1+x+xy$	$1+x$
	Others	1	$1+x+xy$	$1+x+xy$
R.C.	Q.M.	1	xy	xy
	D.D.	$1+y$	$1+x+xy$	$1+x$
	Others		N/A	
W.C.	F.O.	2	$2x+xy$	$x+xy$
	R.C.	2	$x+2xy$	$x+xy$
	W.C.		N/A	
	Q.M.	1	xy	xy
	D.D.	$1+y$	$1+x+xy$	$1+x$
Q.M.	Any		N/A	
D.D.	R.C.	$1+x$	$2xy$	$1+xy$
	D.D.	$1+\max(x,y)$	$2x+xy$	$1+x$
	Others	$1+x$	$1+x+xy$	$1+xy$

4 CAMP Attacks

Existing application-layer amplification attacks [5, 26, 29, 11] exploit *individual* amplification primitives covered by our taxonomy. However, the amplification potential of DNS goes far beyond what can be achieved by these primitives in isolation. In this section, we analyze the composability of amplification primitives and introduce a family of ***Compositional Amplification*** (CAMP) vulnerabilities, which combine multiple primitives to produce multiplicative amplification effects. We first examine all possible regular compositions of two primitives with concrete examples (Section 4.2), and then explain the extension to arbitrary compositions (Section 4.3). Finally, we discuss how an attacker can leverage these building blocks to launch real (D)DoS attacks on DNS servers (Section 4.4).

4.1 Adversary Model

We consider an attacker that attempts to exhaust a DNS server's communication or computational resources at disproportionate attack cost. The target can be a resolver or a nameserver, or both simultaneously, depending on the attack setting. While a resolver is necessarily an

```
>zone a.com@1.2.3.4
q.a      NS   n1.a
q.a      NS   n2.a
q.a      NS   n3.a

n1.a     NS   n11.b
n1.a     NS   n12.b
n1.a     NS   n13.b
n2.a     NS   n21.b
...
>zone b.com@1.2.3.4
...
```

(a) Fan-out as secondary

```
>zone a.com@1.2.3.4
q.a     NS     n1.a
q.a     NS     n2.a
q.a     NS     n3.a
-v1: secd. referral
n1.a    NS     n11.b
n2.a    NS     n21.b
n3.a    NS     n31.b
-v2: secd. rewrite
n1.a   CNAME   r11.b
n2.a   CNAME   r21.b
n3.a   CNAME   r31.b
```

```
>zone b.com@1.2.3.4
-v1
n11.b    NS     n12.b
n12.b    NS     n13.b
n21.b    NS     n22.b
n22.b    NS     n23.b
...
-v2
r11.b   CNAME   r12.b
r12.b   CNAME   r13.b
r21.b   CNAME   r22.b
...
```

(b) Chaining as secondary (two versions)

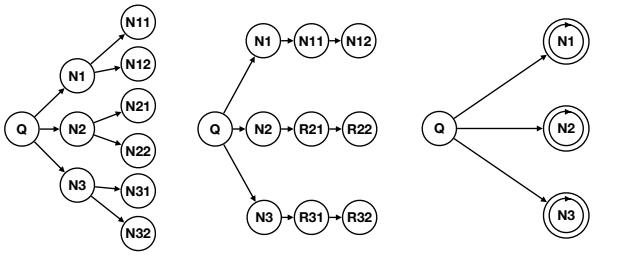
```
>zone a.com@1.2.3.4
-v1: secondary as QMIN
q.a    NS    l3.l2.l1.n1.b
q.a    NS    l3.l2.l1.n2.b
q.a    NS    l3.l2.l1.n3.b

>zone b.com@1.2.3.4
-v1: secondary as QMIN
-Any RR type can work
l3.l2.l1.n1.b   A 5.6.7.8
l3.l2.l1.n2.b   A 5.6.7.8
l3.l2.l1.n3.b   A 5.6.7.8
```

(c) Self-probing as secondary

Figure 2: Example DNS zone configurations for compositions with the primary primitive being **fan-out**.

initiator and a potential victim for amplification, the excessive queries it generates can also be directed towards a victim nameserver. In line with prior studies [5, 11], we assume that the attacker can install zone files crafted for amplification on a *focal* nameserver, which is supposed to receive most queries and, if necessary, also on other auxiliary servers. The focal nameserver is not necessarily the target of an attack, and it can be provided by a DNS hosting service. The attacker should register the malicious zones' NS and glue records in the corresponding parent zones, e.g., through a domain registrar. The attacker is not required to access the involved nameservers in any other way.



(a) **Fan-out** × Fan-out (b) **Fan-out** × Chaining (c) **Fan-out** × Self-probing

Figure 3: Illustration of compositions with the primary primitive being fan-out. Each arrow indicates an RR that maps one name to another. For (b), we mix the cases where the secondary chain is either referral or rewrite.

4.2 Composability Analysis

The number of possible ways to arbitrarily compose amplification primitives may appear daunting at first sight. To start with, we narrow down the scope of our analysis to those *regular* compositions that produce a *multiplicative* effect. In a regular multiplicative composition of two primitives (denoted as **primary × secondary**), each derivative of the primary serves as the base of a separate instance of the secondary. Some compositions are easy to construct and comprehend (e.g., when the primary is fan-out), whereas others are less so and some are altogether impossible to construct. The resulting amplification effect also varies.

Table 2 summarizes the results of our composability analysis. We do not distinguish the two variants of fan-out because they have identical construction (glueless or out-of-bailiwick NS) and they can co-occur with an initial batch of concurrent queries followed by sequential queries for NS names. This leaves us 5 amplification primitives to analyze, and we identify 16 (out of 25) combinations that lead to meaningful and valid zone setups. All of them produce multiplicative amplification on a resolver and 12 of them on a nameserver.

Our analysis proceeds in the following unified setting, which helps to illuminate the commonalities and nuances of different compositions. For space reasons, we provide examples for some of the compositions in this section and leave the others to Appendix A.

Analysis Setup. We analyze the amplification effect on a resolver and a focal nameserver triggered by a single client request. When a composition's construction needs auxiliary nameservers, the number of them should be kept to a minimum for cost efficiency and ease of management. The focal nameserver's IP is assumed to be `1.2.3.4` and the auxiliary nameservers be `5.6.7.8` or `x.x.x.x`.

We always configure the primary primitive under the SLD `a.com` and the secondary under `b.com`, except in some cases involving QMIN. For simplicity, all names in our example zone files are written *in a shortened form relative to the TLD*, e.g., a fully qualified name `x.a.com.` is displayed as `x.a` without the trailing dot. An NS name starts with '`n`' and an alias (i.e., the owner of CNAME) with '`r`'. A set of names is denoted, for example, by `n[x]` or `n[xy]`, where `x,y` $\in \{1, 2, \ldots\}$. Unless otherwise specified, the client QNAME is always `w.q.a.com`.

We assume that the resolver already caches (glued)

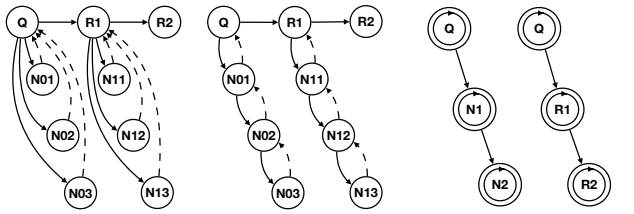
```
>zone a.com@1.2.3.4 |>zone b.com@1.2.3.4
q.a   NS   n01.b    |n01.b   A   5.6.7.8
q.a   NS   n02.b    |n02.b   A   5.6.7.8
q.a   NS   n03.b    |n03.b   A   5.6.7.8
                    |n11.b   A   5.6.7.8
r1.a  NS   n11.b    |n12.b   A   5.6.7.8
r1.a  NS   n12.b    |n13.b   A   5.6.7.8
r1.a  NS   n12.b    |...
                    |>zone q.a.com@5.6.7.8
r2.a  NS   n21.b    |w.q.a   CNAME  w.r1.a
r2.a  NS   n22.b    |>zone r1.a.com@5.6.7.8
r2.a  NS   n23.b    |w.r1.a  CNAME  w.r2.a
```

(a) Fan-out as secondary

```
>zo a.com@1.2.3.4 |>zo q.a.com@5.6.7.8
q.a    NS  n01.b  |w.q.a   CNAME   w.r1.a
r1.a   NS  n11.b  |>zo r1.a@5.6.7.8
r2.a   NS  n21.b  |w.r1.a  CNAME   w.r2.a
>zo b.com@1.2.3.4 |>zo r2.a@5.6.7.8
n01.b  NS  n02.b  |w.r2.a  CNAME   w.r3.a
n02.b  NS  n03.b  |...
n03.b  A  5.6.7.8 |
n11.b  NS  n12.b  |>zo n[x]1.b.com@5.6.7.8
n12.b  NS  n13.b  |n[x]1.b   A   5.6.7.8
n13.b  A  5.6.7.8 |>zo n[x]2.b.com@5.6.7.8
...               |n[x]2.b   A   5.6.7.8
```

(b) Chaining (referral) as secondary

Figure 4: Example DNS zone configurations for compositions with the primary primitive being **chaining** (rewrite).



(a) **Chaining** × Fan-out (b) **Chaining** × Chaining (c) **Chaining** × Self-probing

Figure 5: Illustration of compositions with the primary primitive being chaining. A dashed arrow always starts from an NS name, indicating that the corresponding nameserver provides records for the name pointed by the arrow. We have four possible versions for (c): {rewrite chain, referral chain} × {QMIN, DDLG}.

referrals for `a.com` and `b.com` so that it can directly reach their nameservers, but no record for other names under these zones is cached prior to the client request.

4.2.1 Fan-out as Primary

When used as the primary primitive, fan-out can be composed with any other primitive, as illustrated in Figure 3. Each derivative of the primary fan-out can elicit another fan-out or a chain, or be a deep name triggering self-probing. Figure 2 show example zone configurations for all these compositions except the secondary being DDLG. All of them require just one nameserver. To answer a client request, a resolver first issues one query for `w.q.a.com`, followed by queries for `n[x].a.com` and `n/r[xy].b.com`, or by QMIN queries for `l3.l2.l1.n[x].b.com`. All these queries are directed to the focal nameserver.

For **fan-out × fan-out**, the content of zone `b.com` is unimportant, but its delegation in zone `com` should be set properly to ensure that queries for the secondary primitive's derivatives `n[xy].b` arrive at the focal nameserver. Another subtlety is that some DNS implementations may signal in-bailiwick yet glueless NS RRs as misconfiguration [21] and reject the zone files. To avoid this issue, we can change `n[x].a` to `n[x].b` and `n[xy].b` to `n[xy].c`, without affecting the amplification effect.

The two versions of **fan-out × chaining** are similar, as shown side-by-side in Figure 2b. For the referral chain version, the aforementioned issue of missing glue may arise as well, but this can be resolved again by distributing the chaining NS names to different SLDs all hosted by the focal nameserver. For the rewrite chain version, we apply the construction based on a single zone (Section 3.2.2). In case a resolver does not implement strict response sanity checking, one can resort to an Unchained-style setup with an auxiliary nameserver [11].

Figure 2c gives an example zone configuration for **fan-out × self-probing** with QMIN being the secondary primitive. This composition is simpler to configure than the others, because a single record with a deep name suffices to induce multiple probing queries, and the type of such records does not matter as long as they exist.

The other version **fan-out × DDLG** requires extra servers to set up. It causes multiplicative amplification only on a resolver but not on a single nameserver because the queries are spread out.

4.2.2 Chaining as Primary

A referral chain is quite different from a rewrite chain when treated as the primary primitive in a composition.

Referral Version. Composing a referral chain with a secondary fan-out or chaining primitive is impossible, because this breaks the primary primitive's semantics and prevents the composition's regularity and multiplicative amplification. For example, imagine that the first derivative of the primary spawns multiple NS names, then all of them should be regarded as the derivatives of a fan-out instance, thus stopping the primary chain

```
-The client query is l3.l2.l1.q.a.com
>zo a.com@1.2.3.4       |>zo b.com@1.2.3.4
q.a          NS  n01.b  |n01.b  A  0.0.0.0
q.a          NS  n02.b  |n02.b  A  0.0.0.0
q.a          NS  n03.b  |n03.b  A  0.0.0.0
>zo q.a.com@0.0.0.0     |
l1.q.a       NS  n11.b  |n11.b  A  1.1.1.1
l1.q.a       NS  n12.b  |n12.b  A  1.1.1.1
l1.q.a       NS  n13.b  |n13.b  A  1.1.1.1
>zo l1.q.a.com@1.1.1.1 |
l2.l1.q.a    NS  n21.b  |n21.b  A  2.2.2.2
l2.l1.q.a    NS  n22.b  |n21.b  A  2.2.2.2
l2.l1.q.a    NS  n23.b  |n21.b  A  2.2.2.2
```

Figure 6: Example for **self-probing** (DDLG) × fan-out.

from growing further[1]. When the secondary is also a referral chain, the two primitives would merge into a single chain[2]. When the secondary is a rewrite chain, the exclusiveness of CNAME RR will immediately terminate the primary referral chain at its first derivative.

Nevertheless, composing a primary referral chain with a self-probing primitive is permitted. The case of QMIN is straightforward and produces the expected multiplicative amplification on both resolver and nameserver. Similarly to **fan-out × DDLG**, **referral-chain × DDLG** distributes queries to multiple nameservers and so is applicable to targeting resolver only.

Rewrite Version. A rewrite chain as the primary primitive can be composed with each of the other four primitives, as illustrated in Figure 5. The central idea is to force a resolver to take a "detour" for each name while chasing the rewrite chain. But how would this be possible if the existence of a CNAME RR excludes any other RRs for the name?

We work around this restriction by employing a suffix (`r[x].a`) of each name on the primary rewrite chain, rather than the names themselves, to spawn NS names for a secondary primitive, as shown in Figure 4. Here we provide examples for fan-out and referral chain; the case for DDLG is similar, and the case for QMIN is trivial. Note that the base of a chaining primitive can already spawn a secondary primitive (indicated by index 0). The sequence of queries triggered by a client request will be: `w.q.a.com` → `n0[y].b.com` → `w.q.a.com` → `w.r1.a.com` → `n1[y].b.com` → `w.r1.a.com` → ...

There are a few important technicalities. First, we should host the primary rewrite chain on a different server (i.e., `5.6.7.8`) than the focal server, as otherwise the resolver will directly follow the chain without resolving the secondary NS names. This is required no matter how the construction varies (e.g., using multiple SLDs). Second, the nameserver(s) indicated by these NS names must be resolvable and provide the CNAME RRs to construct the rewrite chain. For the case of fan-out, providing the corresponding A records (all mapping to the same IP) is straightforward, and the resolver is always directed to the same server hosting CNAME RRs. The case of referral chain is trickier: the NS names are queried forward along the chain and they must be resolved in the reverse direction with extra queries. This requires the proper setup of all but the last NS names (i.e., `n[x][y-1].b.com`) in separate subzones.

4.2.3 Self-probing as Primary

QMIN cannot be used as the primary primitive of a composition because, unlike other primitives, it is not driven by a set of RRs crafted to guide the interactions between a resolver and nameservers, but rather by a single RR as a mere resolver-side mechanism. In contrast, DDLG can be composed with any primitives.

The derivatives of DDLG are a sequence of its base name's suffixes with increasing length. Conceptually, its compositions resemble those of a chaining primitive, as depicted in Figure 11. We give an example zone configuration for **DDLG × fan-out** in Figure 6, leaving examples for other cases and the discussion of subtleties to Appendix A. The client query now becomes a deep name, and the sequence of queries will be (assuming QMIN is enabled): `q.a.com` → `n0[y].b.com` → `l1.q.a.com` → `n1[y].b.com` → `l2.l1.q.a.com` ...

4.3 Arbitrary Composition

Our composability analysis enumerates all permissible and distinct ways to combine two amplification primitives in a regular form. The results extend naturally to the composition of arbitrary numbers of primitives. For example, in **fan-out × chaining**, making all names of the secondary primitive deep gives rise to a three-dimensional (3D) composition **fan-out × chaining × QMIN**; creating a chain from each derivative of the secondary in **chaining × fan-out** leads to **chaining × fan-out × chaining**.

Compositions can also be irregular, where a primitive's derivatives create different types of primitives: for example, a fan-out instance's first derivative initiates another fan-out instance, whereas the second initiates a rewrite chain and the third initiates a referral chain. This is illustrated in Figure 3 (b) and Figure 11 (b). Moreover,

[1] One may argue that one of the fan-out NS names can be used as the primary chain's derivative to create another instance of fan-out, and so on. However, this will hinder a consistent definition of regular multiplicative composition.

[2] One may similarly argue that each derivative of the primary can spawn two NS names, one to create a secondary referral chain and the other to grow the primary chain. This will su□er from the same definitional dilemma as in the previous note.

primitives in one dimension need not have a uniform size in terms of the number of derivatives. Their sizes can be adjusted according to the limits implemented by DNS servers. Allowing these irregularities substantially adds to the variety and flexibility of CAMP attacks.

Since regular compositions already achieve large amplification beyond the reach of any individual attack vector, we focus on them in the rest of this paper, leaving a full accounting of arbitrary composition for future work.

4.4 Launching Real Attacks

CAMP attacks build on the composition of amplification primitives. We define an instance of a composition as a group of RRs implementing it; each instance is activated by one triggering query. An attacker can start with a reconnaissance of the involved DNS servers for their vulnerabilities, then deploys the most effective composition instances, and finally sends coordinated queries to trigger the attack. Three types of targets are possible.

Focal Nameserver. At the center of amplification, a focal nameserver is a natural target of attacks. An attacker can employ an array of resolvers and coordinate them to process duplicate queries for composition instances installed at the focal nameserver. The resolvers can be those open ones on the order of millions [23], or the closed ones accessible to the attacker, e.g., through a botnet or Internet measurement probes [1]. An important factor for prolonged attacks is the bypassing of resolver caching. This can be achieved by deploying a large number of composition instances (which would increase the attacker's costs) for queries in rotation and/or setting small TTL values for attack-related RRs.

Arbitrary Nameserver. CAMP attacks can target an *arbitrary* nameserver, including the root and TLD servers, even if the attacker cannot install zone files on it. This is enabled by compositions where the primitives along the last dimension are fan-out. Specifically, we can change the derivatives of these fan-out instances to NS names in any zone hosted by the target nameserver. For example, changing `n[x][y].b.com` to `n[x][y].tld` if the target is a TLD server, or to `ns.n[x][y]` if the target is a root server. In some cases, such as **chaining × fan-out** or **self-probing × fan-out**, one of the NS names should remain under the attacker's control and resolve to a server that provides the necessary RR for the resolver to traverse along the primary primitive.

Resolver. When the target is a resolver, the attacker will need to deploy composition instances on multiple focal nameservers according to the servers' capacities, the expected attack intensity and duration, as well as the algorithm used to coordinate triggering queries.

Table 3: Default query limits implemented by popular open-source DNS resolvers to mitigate DoS risks.

Resolver Limits	BIND 9.18.4	Unbound 1.16.0	PowerDNS 4.7.3
Concurrent NS queries	5	3	1
Failover NS queries	-[1]	3	9
Total NS queries	-	6	10[2]
Referral chain length	7	4	15
Rewrite chain length	17	12	12
QMIN iterations	5	10	10
DDLG iterations	>20	>20	>20
Max queries per cli. req.	100	32	60[3]

1: No explicit limit according to our measurement.
2: Controlled by PowerDNS's `max-ns-address-qperq` parameter. The default value 10 decreases for every additional NS RR in a response to a minimum of 5.
3: Controlled by PowerDNS's `max-qperq` parameter and raised to 100 when QMIN is enabled.

5 Evaluation

We have validated CAMP vulnerabilities in both popular DNS software and public DNS services. The central metric in our measurement is the message amplification factor (MAF), which we define as the number of DNS queries *received*[3] by the *focal nameserver* during the resolution of one triggering client request. Other common metrics include the packet amplification factor (PAF) and the bandwidth amplification factor (BAF). Clearly, the MAF establishes a lower bound on amplification power[4]. Using MAF also aligns with our focus on analyzing application-layer vulnerabilities while abstracting away lower-layer details.

Ethical considerations. For measuring open resolvers (Section 5.2) and attack simulation (Section 5.3), we use our own domains and authoritative nameservers without affecting the public DNS infrastructure, except occasionally sending a few queries to root and TLD servers. Our servers run on cloud VMs. We have confirmed with the cloud provider that our experiments do not violate their user policy nor raise any security alert. The experiments generate small volumes of intermittent queries: even in the most intensive simulated attack, no more than 200 queries per second (QPS) on average are sent by each resolver within half an minute. The query load is negligible compared with their usual workloads (QPS in the millions or higher), and all attack-related RRs have a low TTL of 5 seconds. Therefore, our experiments have no impact on the public resolvers' normal operation.

[3] Some studies count both traffic sent and received by a server [5].
[4] For DNS amplification attacks, the PAF can be 5× larger than the MAF if TCP fallback is factored in [5], and the BAF can be even higher if, for example, DNSSEC is enabled.

Table 4: The MAF of CAMP compositions measured with resolver software. The composition instances are configured according to the default limits of the resolvers (Table 3), with several exceptions explained in the notes below. A shaded cell indicates that the composition fails to produce a multiplicative amplification effect.

Primary	F.O.				W.C.			R.C.	D.D.				F.O.	W.C.	D.D.
Secondary	F.O.	R.C.	W.C.	Q.M.	F.O.	R.C.	Q.M.	Q.M.	F.O.	R.C.	W.C.	Q.M.	W.C.	F.O.	W.C.
Tertiary	-	-	-	-	-	-	-	-	-	-	-	-	Q.M.	R.C.	Q.M.
Compo. Index	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*
BIND	31	36	21	21	**119**	**136**	82	8	80	50	2	21	26	**731**	2
Unbound	12	17	73	61	28	60	**112**	43	30	67	**241**	**201**	**726**	23	**2400**
PowerDNS	57	57	56	91	24	31	99	98	21	30	53	90	97	11	97

- For BIND, we set the size of fan-out instances to 5 in attacks *a* and *b*, because it does not send failover queries for non-existing NS names; we increase the size to 20 in the other attacks *e, i* and *m* where failover NS queries are triggered.
- The deep names of QMIN instances contain 16 labels after the TLD, e.g., `15.14...1.a.com`. This ensures that all three resolvers, which implement different QMIN algorithms, reach their iteration limits. We disable their QMIN feature for compositions not involving it.
- None of the resolvers explicitly restricts the number of DDLG iterations, which seems to be restricted only by the global query limit. We choose a value of 20 for a balance between setup complexity and comparability with other primitives.
- We disable AAAA queries (IPv6 addresses) for NS names in all three resolvers to obtain a better picture of the multiplicative amplification effects. More queries will be generated if this feature is enabled.

5.1 Vulnerabilities in DNS Software

DNS resolvers play a central role in CAMP vulnerabilities because they drive the name resolution process, whereas nameservers answer queries passively. We examine three industry-standard resolver implementations: BIND in the recursive mode, Unbound, and PowerDNS recursor. For each of them, we choose a recent stable version that has been patched in response to the NXNS attack [5].

Methodology. We develop a Docker-based testbed to facilitate our evaluation and for the reproducibility of our results. It allows the flexible simulation of a realistic DNS infrastructure. All DNS servers and clients run in separate containers. Our architecture comprises: two nameservers for a customized root zone and TLD, plus the necessary number of nameservers for attacker-controlled zones, all running BIND in the authoritative mode; one resolver using one of the three implementations; and one client to generate triggering queries.

We focus on evaluating all 12 of the 16 regular 2D compositions that produce multiplicative amplification on the focal nameserver (i.e., those not employing DDLG as the secondary primitive), and 3 representative regular 3D compositions, each using different primitives.

Measurement Results. Our results are summarized in Table 4, *where each composition is assigned an index for brevity*. Overall, every composition results in multiplicative amplification for at least one resolver, with the measured MAF matching the expected value.[5] The strength of compositions varies across resolvers. While there is no clear winner, compositions involving the rewrite chain and QMIN tend to be the most powerful. Below we highlight our main observations from the evaluation, leaving further explanation to Appendix B.

O1: CAMP can bypass individual query limits. DNS resolvers have implemented various query limits (Table 3) to bound resource consumption and hence reduce the risk of DoS attacks. While effective for known attacks that exploit individual amplification primitives, these limits fail to address CAMP: a composition's MAF can far exceed individual limits. A global quote on resolver queries per client request would serve as a safety net to curb the overall effect caused by any application-layer amplification compositions. Yet a correct implementation is non-trivial: among the three resolvers, only PowerDNS gets this right (which is why it never reaches an MAF over 60, or 100 when QMIN is enabled), whereas both BIND and Unbound reset the corresponding counters after each query rewrite or referral.

O2: CAMP can grow exponentially. The results for 3D compositions demonstrate that the amplification power of CAMP can grow exponentially in the number of dimensions. In the case of Unbound, the tertiary primitives contribute a perfect multiplicative factor to the MAFs of compositions *m* and *o*. For a given resolver, it is possible to find the multi-dimensional, possibly irregular, composition that produce the highest MAF. This is especially alarming in the absence of a correctly implemented global query limiting mechanism.

O3: RFC compliance is critical but complicated. While it remains controversial among DNS implementations whether an NS name can own a CNAME/DNAME RR, it is explicitly stated in RFC 2181 [16, §10.3] that *"The domain name used as the value of an NS record, or part of the value of an MX record must not be an alias"*. BIND complies with this rule, protecting itself from

[5] Note that to calculate the expected MAF of a composition subject to a resolver query quota, we must account for queries sent to other nameservers in addition to the focal nameserver.

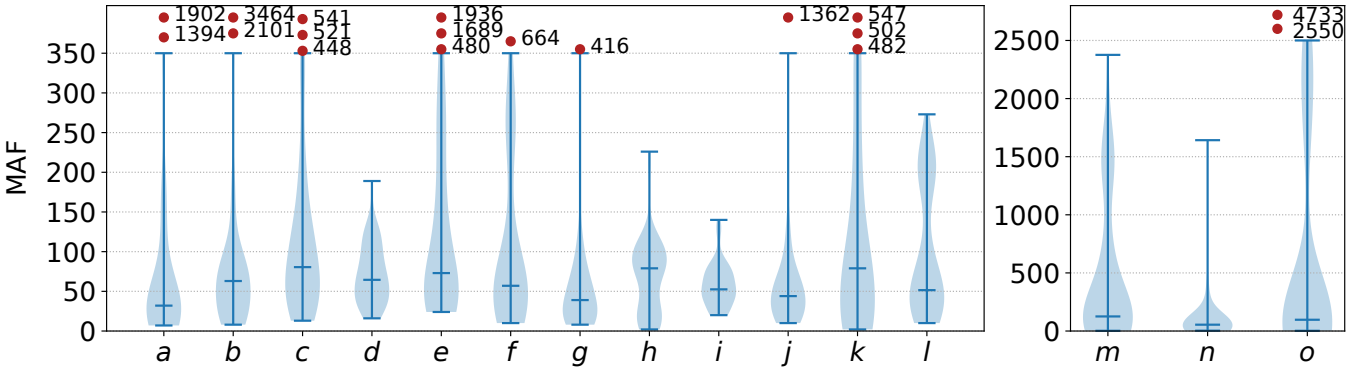


Figure 7: CAMP vulnerabilities measured on 60 public resolvers. Outliers with high MAFs are marked as red dots above the violin plots. The MAFs of most resolvers remain consistent across our measurements over a 10-month period, but some resolvers exhibit high variance from time to time (discussed in Section 5.2) and some seem to have applied security patches after our disclosure. The data reported above is collected between 2023-11-22 and 2023-11-29.

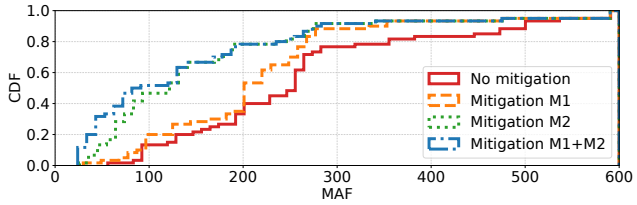


Figure 8: Distributions of the highest MAFs achieved by 2D compositions on 56 public resolvers. The solid red line represents our original measurement data. Other lines hypothesize the cases where all resolvers implement the mitigation options M1 and/or M2 (Table 5) and thereby become resistant to certain compositions.

compositions *c* and *k*. In contrast, the more permissive implementation by Unbound and PowerDNS makes them more vulnerable. Nonetheless, RFCs are not always clear, which leaves room for interpretations that cause problems. One particularly relevant example is whether QMIN should be applied to resolver queries generated during the resolution of a client request [8]. All three resolver implementations perform QMIN for rewritten names (and two of them also for NS names) and therefore increase their attack surfaces. It is important to resolve such ambiguities in RFCs with a joint effort from Internet standardization bodies and DNS developer communities.

5.2 Vulnerabilities of Public Resolvers

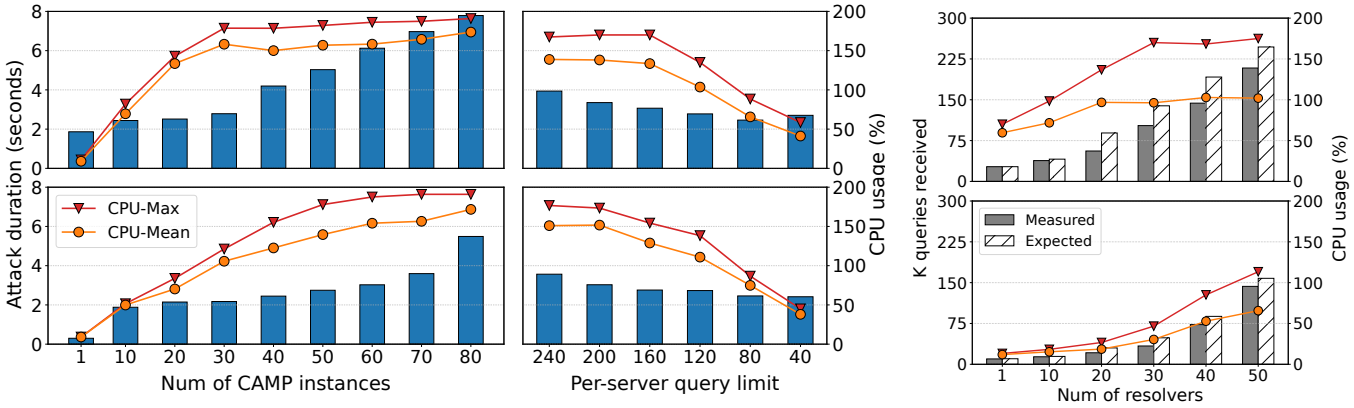
Millions of public resolvers exist on the Internet [19]. Many of them can be abused or targeted by CAMP attacks. We extend our measurements from our local testbed to the real world, analyzing the susceptibility of 60 popular public resolvers (details in Appendix B).

Methodology. Unlike in our local setup, here we deploy attack-related RRs under our registered SLDs to our own public-facing nameservers running on cloud VMs. This allows public resolvers to communicate with the nameservers when receiving our triggering client requests. It is difficult to obtain the precise amplification-related query limits set by public resolvers. For ease of management, we use a set of default parameters for all compositions: 30 for fan-out, 26 for chaining, 16 for QMIN and 20 for DDLG. Some resolvers terminate resolution prematurely due to long referral or rewrite chains; in these cases, we reduce the chain length to a maximum value that allows the resolvers to continue. Nonetheless, our choice of parameters still produces underestimated MAF because premature termination may happen for other reasons and some resolvers indeed have higher query limits.

Our measurements are further complicated by several factors. Large public DNS resolver systems have complex internal architectures, e.g., multi-layer caching and load balancing [32]. They normally accept client requests at an ingress IP and employ multiple independent or collaborating egress resolvers to perform iterative resolution with authoritative nameservers. We have also observed shared egress resolvers used by different ingress resolvers and sporadic unsolicited queries to our nameservers. To reduce the noise in and the interference between measurements, we measure each pair of resolver and composition sequentially with a long cooldown period and estimate the MAF by counting at our nameservers the queries from all egress resolvers mapped to the ingress resolver.

Measurement Results. Figure 7 depicts the distribution of MAFs produced by each composition. Overall, all compositions are effective on multiple resolvers to varying extents, and 3D compositions can achieve higher MAFs than 2D compositions.

O4: Some resolvers are exceptionally vulnerable. We have observed multiple abnormal cases with exceptionally high MAFs, e.g., over 1000 for 2D compositions.



(a) Attacks using 10 closed resolvers. The two subplots on the left are for attacks with varying numbers of composition instances and the resolvers' per-server rate limiting disabled. The two subplots on the right are for attacks with 80 instances and varying `fetches-per-server` values set for the BIND resolvers. The attack's duration is shown in bars and the target server's (dual-core) CPU usage in lines.

(b) Attacks using varying numbers of open resolvers. The number of expected and actual queries received by the target server during an attack is shown in bars and the server's max and mean CPU usage in lines.

Figure 9: CAMP attack simulation results under different settings. The three upper subplots are for attacks targeting the focal nameserver. The three lower subplots are for attacks targeting an arbitrary nameserver. All attacks use composition `e` except that those reported in the upper subplot of (b) use `c`. Each upper subplot shares the x-axis with the corresponding lower subplot. All the four subplots in (a) share the same dual y-axes shown on the two sides.

There are several reasons behind this: (1) extra queries of type AAAA (and in some cases the obsolete type A6 [20]) for IPv6 addresses when resolving NS names; (2) TCP fallback—even after the DNS Flag Day 2020 where the maximum UDP response size for DNS is suggested to be 1232 bytes, many resolvers still stick to the original 512 bytes and switch to TCP when receiving large responses generated by some composition instances; (3) query retries due to timeout and SERFAIL responses, especially when the resolvers do not limit the number of failover queries for NS names. These findings further highlight the complexities involved in thoroughly analyzing DNS amplification vulnerabilities.

O5: Diversity means high risks. A resolver's risk to CAMP is determined by the composition with the highest MAF that it is vulnerable to. This cannot be seen from Figure 7, where the majority of compositions seem to produce an MAF below 100. We show a clearer picture in Figure 8, which depicts the resolvers' risk distribution with respect to the regular 2D compositions. Focusing on the solid red line, we can see that over 80% of the resolvers experience an MAF of over 100 and 60% over 200; the MAF will be higher if 3D and arbitrary compositions are factored in. These high risks result from the the variety of CAMP vulnerabilities. Moreover, the large variance in our measurement data reflects the diverse implementations and configurations of public resolvers. This diversity poses a practical challenge for countering the threat of CAMP at Internet scale.

5.3 Attack Simulation

In addition to individual compositions, we also evaluate the efficiency of real CAMP attacks by simulating them against our own public nameservers.

Methodology. We use small cloud VMs, each of which runs Ubuntu 22.04 with 2 dedicated Intel Xeon 8168 CPUs and 4GB of RAM, for the servers so that the dynamic change of their resource usage can be easily detected even under light workloads. The VMs have a network throughput limit of 2Gbps, and our DNS traffic is not shaped or filtered out by the cloud platform. We also tune the servers' networking buffers to ensure that no packet is dropped by the network interface or kernel throughout our experiments.

We consider two attack settings: the first employs 10 closed BIND resolvers (hosted by us on the same cloud platform as the nameservers) to understand the attack behavior in a controlled environment; the second employs up to 50 open resolvers to illustrate the real-world threat of CAMP. For the former case, we add an artificial delay of 30 ms to the nameservers' egress packets, simulating a typical network condition of authoritative DNS services [3]. For the latter, we use a fixed number of 50 composition instances for each attack.

We analyze two types of targets to demonstrate the versatility of CAMP: a focal nameserver (using composition `c` and `e`), and an arbitrary nameserver (using composition `e`; see Section 4.4). For the latter case, we assign 2/3 of the secondary fan-out primitives' NS names as

Table 5: Summary of mitigation mechanisms for CAMP attacks. In the 4th column, "Res" is short for resolver and "NS" for nameserver. In the last column, "B" is short for BIND, "U" for Unbound, and "P" for PowerDNS (and its associated load balancer dnsdist); a single letter without any symbol means the software has already implemented the mechanism before our disclosure to the vendor, a "!" symbol means the software's prior implementation is flawed in that it can be bypassed by CAMP, and a "*" symbol means the mechanism has been experimentally implemented or is under consideration by the vendor after our disclosure.

Category	Mitigation mechanism	Effective for CAMP	Change to	Adopted by
Protocol-wide patches	M1: Prohibit the rewrite of NS names	*c*, *k*	Res/NS	B, U*, P*
	M2: Restrict QMIN to original client queries	*d*, *g*, *h*, *l*	Res	B*, U*, P*
	M3: Enable QMIN only for the root and TLD	*d*, *g*, *h*, *l*	Res	B*, U*, P*
Holistic query limits	M4: Global quota on resolver queries per client request	All	Res	B!, U!, P*
	M5: Correlated limits on referral depth and width	*a*, *b*, *i*, *j*	Res	P!
	M6: Correlated limits on interacting referral and rewrite	*c*, *e*, *f*, *k*	Res	-
System-wide rate limiting	M7: Max outbound pending queries per nameserver/zone	All	Res	B, U, P*
	M8: Max inbound pending queries per source IP/AS	All	Res/NS	B, U, P
Anomaly detection	M9: Actively scan and remove suspicious zone files	All	NS	-
	M10: Actively monitor suspicious query pattern	All	Res/NS	P

non-existing to the zone hosted by the target server, while keeping 1/3 of the NS names resolvable and the corresponding nameserver reachable to ensure that the resolvers can chase the primary rewrite chains.

For each attack simulation, we use a single client to send a batch of concurrent triggering requests, one for each CAMP instance, to the involved resolvers, while monitoring the victim nameserver's performance metrics.

Simulation Results. DNS servers are compute-intensive, and we observe low memory consumption (< 5%) and network I/O (< 28K packets per second) in all the simulated attacks. Therefore, we choose the victim's CPU usage as the main indicator of attack efficiency, among other metrics like attack duration and query volume. The results are reported in Figure 9.

O6: CAMP attacks can scale up and out. As shown by the two subplots of Figure 9a on the left, with a growing number of composition instances, the attacks' intensity increases linearly until the victim reaches its computational capacity, and afterwards the victim maintains high CPU usage for an increasingly longer period of time. Comparing different targets, we can see that the attacks consume more resources on the focal nameserver than the arbitrary nameserver, which receives roughly 1/3 fewer queries according to our setup, as expected.

For the evaluation on open resolvers shown in Figure 9b, we can see similar attack scaling behaviors as the number of resolvers grows. We also find that the queries received by the victim during an attack are fewer than the expected amount estimated from the MAF of individual composition instances. This can be attributed to resolver-side mechanisms, e.g., the termination of client requests on timeout or rate limiting, since we do not observe any packet drops at our servers or in the cloud network.

Together, the simulation results demonstrate the vertical and horizontal scalability of CAMP attacks. We can estimate the client-side cost to overwhelm a large nameserver with 8 CPUs and 16 GB of RAM [13]. For example, assuming the victim is the arbitrary nameserver, the attacker's client only needs to sends around 20K queries every 25 seconds[6]. This is well within the capability of a personal computer.

O7: Rate limiting helps to mitigate (parallel) attacks. Resolvers often implement per-entity rate limiting mechanisms to mitigate DoS attacks in general. The entity can be a source IP, a server, a zone, etc. We evaluate how BIND's `fetches-per-server` feature (disabled by default) affects the attacks in the closed setting. It limits the number of in-transit queries a BIND resolver sends to each nameserver. The results are shown in the two subplots of Figure 9a on the right. As the limit drops, we observe a gradually reduced load on the victim server. This indicates the effectiveness of `fetches-per-server`. It can also alleviate *parallel attacks* that target or exploit overlapping nameservers, which is similar to our simulated attacks concurrently querying multiple composition instances installed on a single nameserver. One caveat, though, is that an attacker can still *scale out* an attack by employing more resolvers and/or nameservers.

[6] 20K = 50 (instances) 100 (resolvers; 50 2 as per Figure 9b) 4 (hardware scaling factor based on our test server). The attack duration of 25 seconds (not shown in the plot) is long enough for most resolvers to clear cached attack RRs with low TTL values.

6 Mitigation

Mitigating CAMP attacks requires a synergy of mechanisms at different levels, covering the protocol design, implementation, and configuration of DNS, as well as the active monitoring of its runtime behavior. Any mechanism's implications on the system's functionality, performance, and security should be carefully assessed. Table 5 summarizes the mitigation options we have proposed to the relevant entities as part of our disclosure. Some of them have adopted several mechanisms and we are working together to evaluate their patched systems.

Protocol-level patches. Some CAMP vulnerabilities stem from the ambiguities of DNS RFCs. They can be patched if the protocol specifications are revised with precision and implemented correctly.

- **M1**: *Prohibit the rewrite of ns names*, as per RFC 2181 [16, §10.3]. This can be enforced by both nameservers and resolvers, e.g., BIND already signals CNAME's existence for NS names as illegal.
- **M2**: *Restrict QMIN to original client queries* received by resolvers and disable it for any resolver-generated queries, which does not defy the intent of protecting client query privacy.
- **M3**: *Enable QMIN only for the root and TLD*, which would retain the most privacy benefits while minimizing attack surfaces [18]

Holistic Query Limits. We have demonstrated that CAMP can bypass coarsely implemented query limits in unanticipated ways. This necessitates more prudent implementations of holistic and fine-grained query limiting algorithms that cover all corner cases.

- **M4**: *Enforce a global quota for resolver queries per client request* and never reset it for a given request.
- **M5**: *Implement correlated limits on referral depth and width*, e.g., reducing concurrent referral queries allowed for cascading fan-out. Limiting referral depth should consider both the recursive case (referral chain) and the iterative case (dense delegation).
- **M6**: *Implement correlated limits on interacting queries for referral and rewrite*, e.g., lowering the referral depth/width limit when chasing a rewrite chain, or setting varying rewrite depth limits when resolving concurrent NS names.

System-wide Rate Limiting. In addition to the above query limits that cap the MAF of composition instances, DNS servers should also implement rate limiting mechanisms to thrwart *distributed* CAMP attacks that exploit multiple clients and servers, as discussed in Section 5.3.

- **M7**: *Limit outbound queries to each zone and/or each downstream server.* BIND has implemented this with `fetches-per-server` and `fetches-per-zone`. Unbound has a similar option `ratelimit`. PowerDNS is considering adopting this feature.
- **M8**: *Limit inbound queries from each source IP address or AS.* This is a common countermeasure to DDoS attacks, also effective for distributed CAMP attacks especially when the victim is a nameserver.

Anomaly Detection. CAMP attacks exhibit abnormal zone setup and query patterns, rendering themselves detectable by DNS operators.

- **M9**: *Actively scan suspicious zone files* with RRs that can lead to large amplification. Nameservers should reject to load such zone files. This may require collaboration among different operators if an attacker disperses attack-related RRs across multiple zones and nameservers.
- **M10**: *Actively monitor suspicious query patterns* such as repetitive queries for chaining names, or excessive QMIN queries for deep names. This could be done within resolvers/nameservers or by separate monitoring systems such as middleboxes.

E□ectiveness of Mitigation. We re-evaluate the risk of our 60 measured public resolvers as if they have implemented two simple option **M1** and **M2**. The results are depicted in Figure 8. **M1** alone reduces the risk for around 15% of the resolvers with MAFs over 200. **M2** is more effective, achieving substantial MAF reduction for about 40% of the resolvers. The two mechanisms together reduce the risk of half of the resolvers to an MAF lower than 100.

7 Related Work

Reflective DoS attacks that leverage public network services for amplification have posed long-standing threats to the Internet. Rossow analyzed and measured the reflective amplification vulnerabilities of popular UDP-based network protocols including DNS, which can generate large replies with EDNS0 extensions of a size up to 4096 bytes [34]. Researchers also studied such transport-layer DNS amplification risks enabled by DNSSEC [17, 38] and queries of type ANY [37].

Our work focuses on more sophisticated application-layer attacks targeting the DNS infrastructure itself. Prior studies explored individual attack vectors for this purpose [26, 5, 29, 11, 14], as discussed earlier in this paper. We generalize some of them in our taxonomy, and more importantly, systematically examine the composibility of amplification primitives. Researchers also start to inspect implementation-level DoS vulnerabilities in DNS software. The NRDelegation attack [6] drives

up a resolver's CPU load using non-responsive nameservers. The TsuKing attacks make use of non-compliant handling of RD (recursion desired) flag in DNS headers to trigger recursive query loops among resolvers [40]. A promising area of future work is to extend our analytical framework for CAMP to capture implementation aspects, such as computational resources and compliance checks, for more comprehensive analyses.

On the defensive side, Moura et al. [30] investigated how different DNS mechanisms, caching in particular, affect a server's resilience to DDoS attacks. One can draw inspirations from their findings when designing CAMP-resilient resolution protocols. To defend reflective DNS amplification attacks, Herzberg and Shulman [17] designed an authentication system that can filter requests with spoofed IP addresses and identify standard-compliant resolvers based on a challenge-response protocol. Rizvi et al. [33] developed a generic approach to mitigate DDoS attacks through traffic engineering and anycast routing, which is a common technique deployed in the DNS infrastructure especially by the root servers. Bushart and Rossow [12] propose an anomaly detection defense that can filter out the queries generated by an application-layer DNS amplification attack; this is in line with our observation that CAMP attacks can be spotted from abnormal query patterns.

Large-scale measurements of DDoS attacks that target and abuse DNS are crucial to understand the Internet's threat landscape. This is still a largely ongoing endeavor. For example, Sommese et al. [35] find from two longitudinal datasets that the authoritative nameservers for millions of domains were once under attack. Nawrocki et al. [31] report evidence of prevalent reflective DNS amplification attacks by peeking into Internet exchange points; they also observe that those attacks frequently rotate exploitable resolvers. Yazdani et al. [41] characterize the amplification potential of over 2.6M open resolvers, conjecturing that 20% of them contribute to 80% global DNS amplification power; yet their estimation does not consider subtle application-layer vulnerabilities. Moon et al. [28] propose a monitoring service that can continuously quantify the amplification risks of public servers according to malicious query patterns. Our preliminary measurement is the first step towards a comprehensive understanding of the scope and extent of CAMP threats faced by today's Internet.

8 Conclusion

We have developed a novel taxonomical approach to dissect the intricate name resolution process for studying amplification vulnerabilities. Our systematic analysis uncovers a family of fundamental vulnerabilities in DNS that can produce unprecedented amplification effects. We have confirmed their ubiquity and severity in mainstream DNS implementations and popular open resolvers. We also shed light on the effective mitigation of CAMP and the improvements to DNS standards. This work will hopefully set a milestone for improving the crucial Internet naming infrastructure's resilience to DoS attacks.

Acknowledgments

We thank the anonymous reviewers and our shepherd for their valuable feedback. We gratefully acknowledge support from ETH Zürich, ZISC, and from SNSF for project RHINE (200021_215318).

References

[1] RIPE Atlas. https://atlas.ripe.net/.

[2] CVE-2023-4236. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-4236, 2023.

[3] DNS Performance Analytics and Comparison. https://www.dnsperf.com, 2023.

[4] What is a low and slow attack? https://www.cloudflare.com/en-gb/learning/ddos/ddos-low-and-slow-attack/, 2023.

[5] Yehuda Afek, Anat Bremler-Barr, and Lior Shafir. NXNSAttack: Recursive DNS Ine□ciencies and Vulnerabilities. In *Proceedings of the USENIX Security Symposium*, 2020.

[6] Yehuda Afek, Anat Bremler-Barr, and Shani Stajnrod. NRDelegationAttack: Complexity DDoS attack on DNS Recursive Resolvers. In *Proceedings of the USENIX Security Symposium*, 2023.

[7] Gautam Akiwate, Mattijs Jonker, Ra□aele Sommese, Ian Foster, Geo□rey M. Voelker, Stefan Savage, and KC Cla□y. Unresolved Issues: Prevalence, Persistence, and Perils of Lame Delegations. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2020.

[8] S. Bortzmeyer, R. Dolmans, and P. Ho□man. DNS Query Name Minimisation to Improve Privacy. RFC 9156 (Proposed Standard), November 2021.

[9] S. Bortzmeyer and S. Huque. NXDOMAIN: There Really Is Nothing Underneath. RFC 8020 (Proposed Standard), November 2016.

[10] Jonas Bushart. Optimizing Recurrent Pulsing Attacks using Application-Layer Amplification of Open DNS Resolvers. In *Proceedings of USENIX Workshop on O ensive Technologies (WOOT 18)*, 2018.

[11] Jonas Bushart and Christian Rossow. DNS Unchained: Amplified Application-Layer DoS Attacks Against DNS Authoritatives. In *Proceedings of International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2018.

[12] Jonas Bushart and Christian Rossow. Anomaly-based Filtering of Application-Layer DDoS Against DNS Authoritatives. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P)*, 2023.

[13] Cisco. Authoritative DNS Capacity and Performance Guidelines, 2022.

[14] Wouter B. de Vries, Quirin Scheitle, Moritz Müller, Willem Toorop, Ralph Dolmans, and Roland van Rijswijk-Deij. A First Look at QNAME Minimization in the Domain Name System. In David Cho□nes and Marinho Barcellos, editors, *Passive and Active Measurement*, 2019.

[15] J. Dickinson, S. Dickinson, R. Bellis, A. Mankin, and D. Wessels. DNS Transport over TCP - Implementation Requirements. RFC 7766 (Proposed Standard), March 2016. Updated by RFCs 8490, 9103.

[16] R. Elz and R. Bush. Clarifications to the DNS Specification. RFC 2181 (Proposed Standard), July 1997. Updated by RFCs 4035, 2535, 4343, 4033, 4034, 5452, 8767.

[17] Amir Herzberg and Haya Shulman. DNSSEC: Security and availability challenges. In *IEEE Conference on Communications and Network Security (CNS)*, 2013.

[18] Scott Hollenbeck and Burt Kaliski. A Balanced DNS Information Protection Strategy: Minimize at Root and TLD, Encrypt When Needed Elsewhere. `https://www.ndss-symposium.org/ndss-paper/auto-draft-130/`, 2021.

[19] Liz Izhikevich, Gautam Akiwate, Briana Berger, Spencer Drakontaidis, Anna Ascheman, Paul Pearce, David Adrian, and Zakir Durumeric. ZDNS: A Fast DNS Toolkit for Internet Measurement. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2022.

[20] S. Jiang, D. Conrad, and B. Carpenter. Moving A6 to Historic Status. RFC 6563 (Informational), March 2012.

[21] Siva Kesava Reddy Kakarla, Ryan Beckett, Behnaz Arzani, Todd Millstein, and George Varghese. GRoot: Proactive Verification of DNS Configurations. In *Proceedings of the ACM SIGCOMM Conference*, 2020.

[22] Aqsa Kashaf, Vyas Sekar, and Yuvraj Agarwal. Analyzing Third Party Service Dependencies in Modern Web Services: Have We Learned from the Mirai-Dyn Incident? In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2020.

[23] Marc Kührer, Thomas Hupperich, Jonas Bushart, Christian Rossow, and Thorsten Holz. Going Wild: Large-Scale Classification of Open DNS Resolvers. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2015.

[24] E. Lewis. The Role of Wildcards in the Domain Name System. RFC 4592 (Proposed Standard), July 2006.

[25] Si Liu*, Huayi Duan*, Lukas Heimes, Marco Bearzi, Jodok Vieli, David Basin, and Adrian Perrig. (*co-first authors). A formal framework for end-to-end DNS resolution. In *Proceedings of ACM SIGCOMM*, 2023.

[26] Florian Maury. The "Indefinitely" Delegating Name Servers (iDNS) Attack. DNS-OARC Spring Workshop, 2008.

[27] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (Internet Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936, 8020, 8482, 8767.

[28] Soo-Jin Moon, Yucheng Yin, Rahul Anand Sharma, Yifei Yuan, Jonathan M. Spring, and Vyas Sekar. Accurately Measuring Global Risk of Amplification Attacks using AmpMap. In *Proceedings of the USENIX Security Symposium*, 2021.

[29] Giovane C. M. Moura, Sebastian Castro, John Heidemann, and Wes Hardaker. TsuNAME: Exploiting Misconfiguration and Vulnerability to DDoS DNS. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2021.

[30] Giovane C. M. Moura, John Heidemann, Moritz Müller, Ricardo de O. Schmidt, and Marco Davids. When the Dike Breaks: Dissecting DNS Defenses During DDoS. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2018.

[31] Marcin Nawrocki, Mattijs Jonker, Thomas C. Schmidt, and Matthias Wählisch. The Far Side of DNS Amplification: Tracing the DDoS Attack Ecosystem from the Internet Core. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2021.

[32] Audrey Randall, Enze Liu, Gautam Akiwate, Ramakrishna Padmanabhan, Geo□rey M Voelker, Stefan Savage, and Aaron Schulman. Tru□ehunter: Cache snooping rare domains at large public DNS resolvers. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2020.

[33] A S M Rizvi, Leandro Bertholdo, João Ceron, and John Heidemann. Anycast Agility: Network Playbooks to Fight DDoS. In *Proceedings of the USENIX Security Symposium*, 2022.

[34] Christian Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS)*, 2014.

[35] Ra□aele Sommese, KC Cla□y, Roland van Rijswijk-Deij, Arnab Chattopadhyay, Alberto Dainotti, Anna Sperotto, and Mattijs Jonker. Investigating the Impact of DDoS Attacks on DNS Infrastructure. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2022.

[36] Sooel Son and Vitaly Shmatikov. The Hitchhiker's Guide to DNS Cache Poisoning. In Sushil Jajodia and Jianying Zhou, editors, *Security and Privacy in Communication Networks*, 2010.

[37] Olivier van der Toorn, Johannes Krupp, Mattijs Jonker, Roland van Rijswijk-Deij, Christian Rossow, and Anna Sperotto. ANYway: Measuring the Amplification DDoS Potential of Domains. In *Proceedings of the International Conference on Network and Service Management (CNSM)*, 2021.

[38] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. DNSSEC and Its Potential for DDoS Attacks: A Comprehensive Measurement Study. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2014.

[39] Zheng Wang. Understanding the Performance and Challenges of DNS Query Name Minimization. In *Proceedings of IEEE International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom)*, 2018.

[40] Wei Xu, Xiang Li, Chaoyi Lu, Baojun Liu, Haixin Duan, Jia Zhang, Jianjun Chen, and Tao Wan. TsuKing: Coordinating DNS Resolvers and Queries into Potent DoS Amplifiers. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2023.

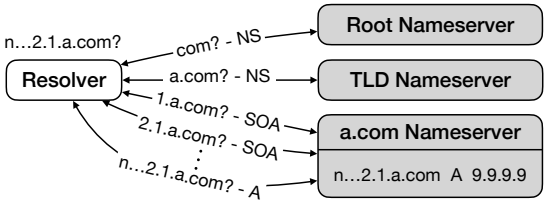
[41] Ramin Yazdani, Roland van Rijswijk-Deij, Mattijs Jonker, and Anna Sperotto. A Matter of Degree: Characterizing the Amplification Power of Open DNS Resolvers. In *Proceedings of the International Conference on Passive and Active Network Measurement (PAM)*, 2022.

A Example Zone Configurations

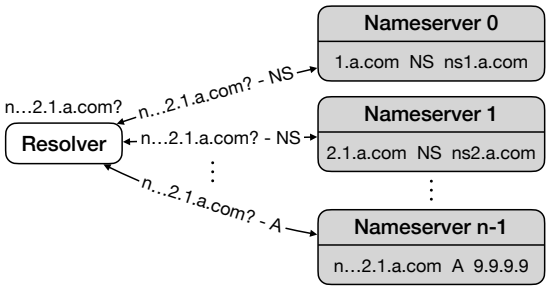
In this section, we give examples of more compositions of amplification primitives and explain subtleties therein.

Fan-out × DDLG: Each derivative of the primary fan-out is a deep name triggering a DDLG instance. The lowest-level zone for each DDLG instance can provide the IP address for the deep NS name (as shown in the example) or not; in either case, a resolver will treat the deep names as resolvable and therefore bypass the MaxFetch(k) limit with extra failover NS queries.

```
> zone a.com @ 1.2.3.4
q.a                NS       l3.l2.l1.n1.b
q.a                NS       l3.l2.l1.n2.b
q.a                NS       l3.l2.l1.n3.b
```



(a) QMIN. The resolver exposes only necessary labels of the client QNAME to the root and TLD nameservers for referrals (simplified for illustration). For the probing queries with more labels, the `a.com` nameserver returns NOERROR responses with SOA records, indicating the existence of empty non-terminals [24] and at least one RR for the deep QNAME.



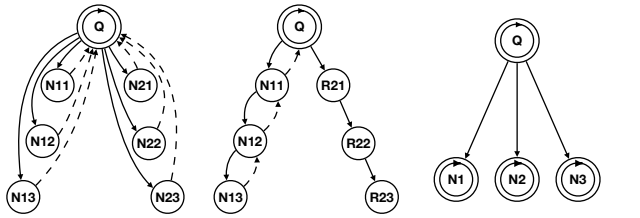
(b) DDLG with a *QMIN-disabled* resolver. The root and TLD nameservers are omitted. The resolver's queries are responded with normal referrals, each by a separate nameserver hosting one zone on the long delegation chain, until the final answer.

Figure 10: Illustration of amplification by a deep name.

```
> zone b.com @ 1.2.3.4
n1.b              NS       n1-l0.b
n1-l0.b           A        0.0.0.0
n2.b              NS       n2-l0.b
n2-l0.b           A        0.0.0.0
n3.b              NS       n3-l0.b
n3-l0.b           A        0.0.0.0
> zone n[x].b.com @ 0.0.0.0
l1.n[x].b         NS       n[x]-l1.b
n[x]-l1.b         A        1.1.1.1
> zone l1.n[x].b.com @ 1.1.1.1
l2.l1.n[x].b      NS       n[x]-l2.b
n[x]-l2.b         A        2.2.2.2
> zone l2.l1.n[x].b.com @ 2.2.2.2
l3.l2.l1.n[x].b A          5.6.7.8
```

Chaining × DDLG: This composition has two versions which are similar. Here the lowest-level zone of each DDLG instance is required to provide a proper NS or CNAME RR for the primary chain to advance.

```
> zone a.com @ 1.2.3.4
-v1: primary as referral chain
q.a               NS       l3.l2.l1.n1.a
n1.a              NS       n1-l0.a
n2.a              NS       n2-l0.a
n3.a              NS       n3-l0.a
-v2: primary as rewrite chain
```



(a) **Self-probing** × Fan-out (b) **Self-probing** × Chaining (c) **Self-probing** × Self-probing

Figure 11: Illustration of compositions with the primary primitive being self-probing. For (b), we mix the cases where the secondary chain is either referral or rewrite.

```
q.a               CNAME    l3.l2.l1.r1.a
r1.a              NS       n1-l0.a
r2.a              NS       n2-l0.a
r3.a              NS       n3-l0.a
-common to both versions
n1-l0.a           A        0.0.0.0
n2-l0.a           A        0.0.0.0
n3-l0.a           A        0.0.0.0
> zone n/r[x].a.com @ 0.0.0.0
l1.n/r[x].a       NS       n[x]-l1.a
n[x]-l1.a         A        1.1.1.1
> zone l1.n/r[x].a.com @ 1.1.1.1
l2.l1.n/r[x].a NS          n[x]-l2.a
n[x]-l2.a         A        2.2.2.2
> zone l2.l1.n/r[x].a.com @ 2.2.2.2
l3.l2.l1.n[x].a   NS   l3.l2.l1.n[x+1].a
l3.l2.l1.r[x].a   CNAME  l3.l2.l1.r[x+1].a
```

DDLG × Chaining: If the secondary primitive is the referral chain, all NS names on each of its instance must be resolved to allow a resolver to reach the corresponding nameservers and traverse along the primary primitive. This composition requires, in addition to the focal nameserver, x servers where x is the DDLG's size. In the example below, the zones `n[x]2.b.com` and `n[x]3.b.com` can be hosted by any of the nameservers for the DDLG zones, i.e., those at IP addresses `x.x.x.x`, instead of the extra nameserver at `5.6.7.8`. For the correct construction of **DDLG × rewrite-chain**, we should use the NS names pointed to by the primary's derivatives, rather than the derivatives themselves, as the bases for instances of the secondary primitive. Note that while RRs of the primary DDLG's derivatives (i.e., suffixes of its base) are hosted by separate servers, they will not trigger extra QMIN probing queries because there is no gap on the deep QNAME's delegation chain.

```
-client query: l3.l2.l1.q.a.com
-common to both versions
> zone a.com @ 1.2.3.4
q.a      NS       n01.b
> zone q.a.com @ 0.0.0.0
l1.q.a   NS       n11.b
> zone l1.q.a.com @ 1.1.1.1
l2.l1.q.a    NS   n21.b
```

```
> zone l2.l1.q.a.com @ 2.2.2.2
l3.l2.l1.q.a    A   9.9.9.9
-v1: secondary as referral chain
> zone b.com @ 1.2.3.4
n01.b    NS       n02.b
n02.b    NS       n03.b
n03.b    A        5.6.7.8
n11.b    NS       n12.b
n12.b    NS       n13.b
n13.b    A        5.6.7.8
...
> zone n[x]1.b.com @ 5.6.7.8
n[x]1.b    A      x.x.x.x
> zone n[x]2.b.com @ 5.6.7.8
n[x]2.b    A      5.6.7.8
-v2: secondary as rewrite chain
> zone b.com @ 1.2.3.4
n01.b    CNAME    r02.b
r02.b    CNAME    r03.b
r03.b    A        0.0.0.0
n11.b    CNAME    r12.b
r12.b    CNAME    r13.b
r13.b    A        1.1.1.1
...
```

Table 6: List of open resolvers used in our measurements

Resolver	IP	Resolver	IP
AdGuard (AG) DNS	94.140.14.14	IP Ex. GmbH	62.146.202.2
AG DNS-Adblock	176.103.130.130	InfoServer GmbH	212.89.130.180
AG DNS-Family	176.103.130.132	Lennart Seitz	94.247.43.254
AG DNS-Unblock	94.140.14.140	Level 3 DNS	209.244.0.3
AliDNS	223.5.5.5	Liteserver	5.2.75.75
Alternate DNS	76.76.19.19	MCI Verizon	195.129.12.122
AMAZON-02	54.93.169.181	Meerfarbig GmbH	95.10.195.195
Baidu Public DNS	180.76.76.76	NTT America	129.250.35.250
Bisping GmbH	62.91.19.67	Neustar	64.6.64.6
Bluewin	83.173.209.124	NextDNS	45.90.30.193
CIRA Canadian	149.112.121.10	Nextgi LLC	134.195.4.2
CNNIC-SDNS	1.2.4.8	Norton-ConnectSafe	199.85.126.10
CenturyLink	205.171.3.65	OVH SAS	217.182.198.203
Cheyenne Tech LLC	80.78.134.11	OnCloud SAS	213.215.11.190
CleanBrowsing	185.228.168.9	OneDNS	117.50.10.10
Cloudflare	1.1.1.1	OpenDNS Home	208.67.222.222
Cogent Comm.	66.28.0.61	OpenNIC	51.77.149.139
Comodo Secure DNS	8.26.56.26	Probe Networks	82.96.65.2
Control D	76.76.2.0	Quad101	101.101.101.101
Cyberlink AG	89.249.44.73	Quad9	9.9.9.9
DNS for Family	94.130.180.225	R-KOM Telekom	81.27.162.100
DNS.WATCH	84.200.69.80	SafeDNS	195.46.39.39
DNSForge	176.9.93.198	ScanPlus GmbH	212.211.132.4
DNSpai	101.226.4.6	Swisscom	195.186.4.110
Deutsche Telekom	194.25.0.68	TEFINCOM S.A.	103.86.96.100
Dyn	216.146.35.35	TREX	195.140.195.21
Fortinet	208.91.112.53	Vodafone	195.27.1.1
Freenom World	80.80.80.80	Yandex DNS	185.184.222.222
GCore Free	95.85.95.85	xTom	77.88.8.8
Google DNS	8.8.8.8	114DNS	114.114.114.114

B More Details on Evaluation

B.1 Measurements on DNS Software

We explain some of our measurement results on popular resolver implementations summarize in Table 4.

Fan-out × Fan-out: The MAF for BIND matches exactly what is expected: 1 (query for the original QNAME) + 5 (queries for the primary NS names) + 5 5 (queries for the secondary NS names). PowerDNS is also vulnerable, but the MAF is capped by its global query limit of 60, and the value 57 is obtained by excluding queries sent to higher-level zones. Unbound aborts the resolution after processing the first batch of 3 concurrent queries for the primary NS names, and for each of them, it sends 1–2 queries for the secondary NS names.

Rewrite Chain × Fan-out: BIND and Unbound are subject to multiplicative amplification but with non-deterministic behavior. After the initial concurrent version of fan-out, BIND sends varying numbers of queries for each subsequent derivative of the primary rewrite chain. This is also the case for Unbound. PowerDNS is somewhat resistant to this attack, resolving only 1 NS per derivative of the primary primitive, but it still produces an MAF of 2 12 = 24. Note that BIND has a global limit per client request of 100, but it gets reset after a query rewrite operation.

Rewrite Chain × Referral Chain: All three implementations are vulnerable. For PowerDNS the measured MAF is lower than predicted because of its global limit and that many queries are sent to the server at `5.6.7.8`.

DDLG × Rewrite Chain: BIND terminates the entire resolution after receiving the first CNAME RR for the base name (`n01.b.com.`) of the first rewrite chain instance. This composition is especially detrimental to Unbound, producing the highest MAF (241) among all 2D compositions. PowerDNS is also vulnerable.

DDLG × QMIN: BIND disables QMIN for NS names and so is not vulnerable. This composition is highly effective on Unbound (201) and PowerDNS (90), which raises it global limit to 90.

Rewrite Chain × Fan-out × Referral Chain: Interestingly, this composition is effective on BIND with a high MAF over 700, as it is vulnerable to both **rewrite chain × fan-out** and **fan-out × referral chain**. PowerDNS's is not vulnerable, which can be explained by its resistance to **rewrite chain × fan-out**. Unbound here deviates from its behaviors in the related 2D compositions; it produces MAF values between 17 and 29 in a non-deterministic way. Since the result does not improve upon its performance in **rewrite chain × fan-out**, we consider this 3D composition ineffective for Unbound.

DDLG × Rewrite Chain × QMIN: BIND is immune to this attack as it prohibits the aliasing of NS names. PowerDNS is vulnerable up to its global query quota. Unbound is exceptionally vulnerable with a remarkable MAF of 2400 (20 12 10).

B.2 Measurements on Public Resolvers

The set of open resolvers used in our measurement study are selected from the union of several sources[7][8][9][10]. We narrow our set down to 60 resolvers that are geographically diverse and remain responsive throughout our measurements, as summarized in Table 6.

[7] https://stats.labs.apnic.net/rvrs
[8] https://www.dnsperf.com
[9] https://publicdnsserver.com/fastest/
[10] https://www.publicdns.xyz