DaLens: Charting DNS Self-Amplification Threats at Large

Liwen Xu
ETH Zurich

Zechao Cai
ETH Zurich

Huayi Duan*
HKUST(GZ)

Adrian Perrig
ETH Zurich

Abstract

The emerging self-amplification attacks (SAAs) pose serious denial-of-service (DoS) risks to the Domain Name System (DNS). They can substantially amplify the interactions between recursive and authoritative servers, depleting resources at disproportionally small costs. Assessing the impact of such attacks on the global name resolution infrastructure is crucial for DNS operators to effectively triage threats and deploy defenses, yet this remains an uncharted and daunting territory.

We have conducted the first large-scale measurement study of SAAs, leveraging a versatile framework, DaLens, which we designed and developed. The work consists of untangling the intricate Open DNS infrastructure to identify effective amplifiers and quantifying their amplification capabilities in a modular, scalable, and sound manner. Out of 307K persistent public resolvers, we find 29K unique resolver clusters that can be exploited in parallel for SAAs, and a significant number of them can still produce large amplification effects even though these vulnerabilities had already been disclosed in prior work.

1 Introduction

The availability of numerous Internet services hinges on the availability of DNS, making this foundational system among the highest-value targets for DoS attacks. A new class of attacks is emerging in DNS's fast-moving threat landscape: they trick recursive resolvers into generating excessive queries for a single client request towards one or more authoritative nameservers, thus enabling an attacker to overwhelm a victim server with a small to moderate number of requests [2, 8, 10, 24, 26, 41]. We refer to them collectively as *self-amplification attacks* (SAAs), for their defining trait of blowing up a typically short name resolution process with intensive interactions between DNS servers themselves.

Given the severity and variety of SAAs, there is a pressing need to assess their impact on the global Open DNS (ODNS) infrastructure. This will enable DNS operators, who often run the services in an altruistic manner [16], to identify vulnerabilities, apply patches, and prioritize defense mechanisms under a constrained budget.

While many methods and tools for DNS measurement exist, we are not aware of any that suit the above goal. Well-known frameworks like MassDNS [35] and ZDNS [17] allow efficient scanning of the DNS namespace to identify misconfigurations. AmpMap [25] can monitor public Internet servers' risk of amplification regarding the size asymmetry of their requests and responses. Several recent measurement-based studies investigate the vulnerabilities of DNS servers subject to conventional reflective amplification attacks [34, 42, 44].

However, the measurement techniques for analyzing SAAs differ fundamentally from existing approaches. In particular, SAAs require the installation of specially crafted records on attacker-controlled nameservers and the coordination of delicate requests to trigger pathological query patterns—and the same applies to benign actors that perform measurements with good intentions. Those query patterns come in a wide variety of forms, many of which require the construction of unconventional DNS zones and the use of non-compliant nameservers. In addition, as the core enablers of SAAs, millions of DNS resolvers form a highly complex fabric whose internal workings are largely unknown, which further complicates the assessment of their amplification capabilities.

To tackle these challenges, we have developed a versatile measurement framework called DaLens and a set of novel measurement methods. At the heart of DaLens is a *programmable reactive nameserver* (PRN) that can dynamically generate answers to DNS queries according to patterns encoded in queries themselves. It allows us to specify and trigger *arbitrary query patterns* in a flexible, cost-efficient, and scalable way, while bypassing restrictions of standard nameservers. Based on that, DaLens offers a comprehensive tool chain to prepare, deploy, and execute measurement tasks on preconfigured probing clients (hereafter, probes), resolvers, and nameservers.

Leveraging DaLens, we have designed and conducted a series of measurements on millions of public resolvers. Our

* Corresponding author: Huayi Duan (huayiduan@hkust-gz.edu.cn)

results have established baseline answers to two essential questions: *(1) Which (clusters of) DNS resolvers are effective amplifiers for SAAs? (2) How capable are they in terms of their achievable message amplification factor (MAF)?* Addressing the first question entails untangling the ODNS infrastructure consisting of different types of resolvers as well as intra- and inter-resolver structure. Addressing the second question requires a systematic exploration of the recursors' configurations and behaviors under diverse adversarial settings.

Over multiple scans of the entire IPv4 address space, we identified over 307K persistent public resolvers. Among them, we find 29K recursors that serve as ultimate amplifiers for SAAs by interacting with nameservers, 110K forwarders that depend on unique upstream resolvers and thus can be exploited to deterministically trigger amplifiers, and 189K forwarders that have multiple upstream resolvers and therefore appear less favorable for attackers. Furthermore, we find that a large portion of amplifiers exhibit lenient query limits, making them particularly vulnerable. Even under moderate amplification patterns, over 10% of them already yield an MAF exceeding 100, with a non-negligible fraction approaching 1000. These are just conservative estimates—many recursors would suffer from even higher amplification if more aggressive query patterns are applied. Such findings are highly concerning, given that many resolvers should have been patched since the disclosure of SAA vulnerabilities [2, 8, 10, 26].

On the upside, our results enable DNS operators to make informed decisions on how and where to prioritize their defense efforts against SAAs. We have disclosed the identified issues to affected operators in a coordinated manner and assisted them in addressing the reported vulnerabilities.

This paper makes the following contributions:

- We formalize the ODNS infrastructure by modeling egress behavior and forwarding dependencies, enabling systematic reasoning about recursive resolution at scale.
- We design a recursor identification methodology that uncovers resolvers' egress sets, classifies recursor types, and derives stable ingress–egress mappings for reliable ODNS structure measurement.
- We develop DaLens, a programmable and scalable SAA measurement framework featuring a reactive nameserver, per-session state tracking, and automated orchestration.
- We conduct the first Internet-scale measurement of SAAs by combining primitive probing with compositional amplification, revealing that the global ODNS ecosystem remains fragile.

Together, these contributions provide a comprehensive methodological and empirical foundation for understanding, quantifying, and mitigating self-amplification threats in today's ODNS ecosystem.

The rest of this paper is organized as follows. We introduce the basics of DNS, its related DoS attacks, and the emerging SAAs in Section 2. We formalize our measurement methodology to address the two core questions above in Section 3, and present the design and usage of DaLens in Section 4. Our measurement results are reported and analyzed in Section 5. We then summarize our coordinated disclosure in Section 6, followed by related work and concluding remarks.

2 Background

2.1 Domain Name System

As a large-scale system for publishing and distributing named information over the Internet, DNS consists of millions of servers that interdepend and interact in complex ways. We distinguish three types of servers:

- an authoritative nameserver (hereafter, simply ***nameserver***) hosts resource records (RRs), which map domain names to IP addresses and various kinds of data, for the zones it is delegated to;
- a recursive resolver (aka ***recursor***) that directly consults nameservers for authoritative answers to name lookup requests from clients;
- a forwarding resolver (aka ***forwarder***) that handles client requests by offloading them to another resolver that is a recursor or potentially another forwarder.

Resolvers cache RRs received from upstream servers to improve lookup efficiency and reduce workload. Answering a request from cache is much faster than processing a cache-miss request [38]. For clarity, we refer to what a resolver receives from its clients as ***requests*** and what it sends to authoritative nameservers as ***queries***. Similarly, we use ***egress*** to denote the IP address from which it issues outbound queries during recursive resolution.

Real-world open DNS infrastructure is not a flat collection of independent resolvers; rather, it exhibits complex interdependencies [5, 36], where forwarders may rely on recursors or even other forwarders, forming multi-hop forwarding chains that obscure the actual entity performing resolution. Furthermore, recursors themselves vary greatly in their egress behavior: some are single-egress, always issuing queries from a single IP address; others are multi-egress, where outbound traffic may originate from different IP addresses under deterministic mapping (e.g., IP-hash load balancing), randomized or round-robin scheduling, or collaborative resolution in which each request is handled by multiple egresses. These dependencies and structural variations complicate efforts to map observed resolver behavior to its underlying structure.

2.2 DNS-related DoS Attacks

DNS is often abused for reflective DoS attacks due to its public nature, lack of authentication, and amplification capability—DNS responses are typically larger than requests

or can be maliciously inflated. Such attacks are becoming less of a concern with the increasing adoption of secure transports for name resolution [12, 14, 15].

DNS servers themselves are also frequent targets of DoS attacks. As with any other Internet service, they can be flooded by direct requests from a large botnet. A slightly more sophisticated form of attack, known as Water Torture [4], generates overwhelming requests for pseudo-random names under a victim domain. As attackers can hide their IP behind resolvers while evading their cache, such attacks are currently widespread. Due to the excessive number of `NXDOMAIN` (non-existing domain) responses, the attack is easy to detect.

Several algorithmic complexity attacks have also been proposed [3, 11] recently. They exploit specific implementation flaws to exhaust a victim resolver's CPU, preventing it from processing normal client requests.

Table 1: Representative attack primitives and their corresponding resolution configurations.

Form	Primitive	Alias	Resolution Configuration Construction
Fan-out	NS Fetch [2, 24]	Fanout	Glueless/Out-of-bailiwick [13] NS RRset
Chaining	NS Chain [24, 26]	Referral	Chaining glueless/Out-of-bailiwick NS RRs
	CNAME Chain [8, 21]	Rewrite	Chaining CNAME RRs
Self-probing	Dense Delegation [10]	DDLG	One NS RR per label of a deep name

Self-Amplification Vulnerabilities. In case of a request's cache miss, a recursor needs to perform the resolution by sending one or multiple queries to the relevant nameserver(s). However, a pathological or maliciously crafted zone configuration can substantially prolong the resolution, causing the recursor to generate a large number of queries. This can be exploited for DoS attacks against DNS servers, and a flurry of such vulnerabilities have been discovered recently [2, 8, 10, 24, 26, 41]. We refer to them collectively as *self-amplification vulnerabilities*, since the firepower of this emerging type of attacks comes from DNS itself and self-inflicts damage.

Following the notion introduced in [10], we categorize their underlying manipulation techniques into a set of attack *primitives*. Each primitive can be realized through a corresponding ***resolution configuration***, meaning a specific arrangement of zone records that induces the resolver to exhibit the targeted behavior. Table 1 summarizes the representative primitives and the resolution configurations that implement them, and concrete examples of these configurations are provided in Appendix A.

Most self-amplification vulnerabilities have their roots in legitimate design choices made by standardized DNS protocols rather than specific implementation bugs. Exploiting the inherent authority delegation mechanism with NS records, an attacker can force a recursor to chase a long delegation chain and thus create an excessive number of queries [24]. Similarly, the essential direction feature enabled by CNAME/DNAME records can be abused to form a long query rewrite chain [8]. It is possible to turn these chains into loops that trap a recursor [26]. Allowing a zone to be delegated to multiple nameservers for availability creates another amplification vector that leverages large sets of NS records [2]. A relatively new feature designed for privacy enhancement, query name minimization (QMIN) [7], also increases a resolver's amplification potential because QMIN may require repeated lookups of a deep name with incrementally longer prefixes, inflating the number of iterative queries a resolver issues. A recent study further examines the composability of individual vulnerabilities and demonstrates multiplicatively larger amplification effects from various compositions [10].

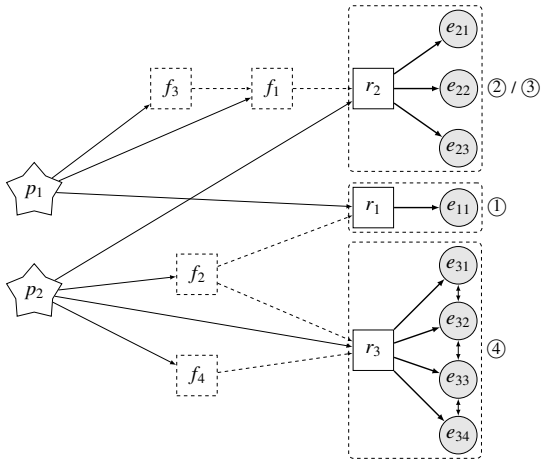


Figure 1: ODNS Architecture. Stars are probes, rectangles are recursors, circles are egresses, and dashed rectangles are forwarders. Solid lines are direct requests; dashed lines denote forwarded requests. Recursor types defined in Section 3.2.1 are labeled ① to ④.

3 Methodology

To chart the global threat landscape of SAAs, we develop a systematic approach that addresses the key challenges outlined in Section 1 and enables comprehensive measurement of the ODNS infrastructure, the ultimate engine of amplification. We begin by formalizing the infrastructure by defining (composite) resolvers' dispatching behavior. We then present a resolver identification pipeline, which extracts per-recursor logic despite caching and forwarding interference. Finally, we apply tailored resolution configurations to uncover the amplification potential of recursors in a systematic manner.

3.1 Formalizing ODNS Architecture

While prior work has discussed components of Open DNS infrastructures descriptively, a formal representation of ODNS entities and their interactions is required to enable systematic reasoning about query forwarding and caching behavior. We

define ODNS as a tuple:

$$O = (S, E, \mathcal{U}, \mathcal{E}),$$

- $S = \{s_1, s_2, \ldots, s_n\}$ denotes the set of all resolvers within the ODNS infrastructure, including both forwarders and recursors.
- E denotes the global set of egresses aggregated across all recursors.
- $\mathcal{U} \subseteq S \times S$ denotes the set of directed forwarding relations among resolvers, where $(s_i, s_j) \in \mathcal{U}$ indicates that resolver s_i forwards requests to resolver s_j.
- $\mathcal{E} \subseteq R \times E$ denotes the set of egress mappings of recursors.

We further define:

- $R \subseteq S$ denotes the set of *recursors*, i.e., resolvers that perform full recursive resolution by directly querying authoritative nameservers. Each $r_i \in R$ is associated with an *egress set* $E_i = \{e_{i1}, e_{i2}, \ldots, e_{in_i}\} \subseteq E$.
- $F = S \setminus R$ denotes the set of *forwarders*, i.e., resolvers that relay incoming requests to one or more upstream resolvers for recursive processing. Each $f_j \in F$ is associated with an *upstream recursor set* $U_j \subsetneq R$.
- $P = \{p_1, p_2, \ldots, p_m\}$ denotes the set of probes (or clients), which send DNS requests q to one or more resolvers in O.
- $O \subseteq S$ denotes the set of *observable resolvers*, which respond to DNS requests on port 53 from our measurement probes.

The structure and dependency among clients, forwarders, and recursors are illustrated in Figure 1.

3.2 Identifying Recursors

In ODNS, recursive resolution is carried out by recursors, with their egresses performing the actual DNS query resolution. In this work, ***we treat a recursor and its associated egress set as conceptually equivalent***, since it is the egress behavior that defines the resolution logic. Accordingly, we refer to the same entity as a recursor when emphasizing its role within the overall infrastructure, and as an egress set when analyzing its observable resolution behavior on a per-query basis.

To identify recursors, we design a three-step methodology: enumerating all egresses to capture the full resolution surface, establishing stable probe–resolver mappings despite load balancing and caching, and determining appropriate minimal sets via egress grouping.

3.2.1 Egress Enumeration

The first step toward reliable recursor identification is to *enumerate all egresses* involved in ODNS resolution. Achieving complete observability is non-trivial due to several factors that can obscure the enumeration process.

The first factor is the *addressing method* of the recursor—specifically, whether it is reachable via unicast or anycast IP addresses; we rely on an existing third-party dataset [1] to identify known anycast deployments and conservatively treat each point of presence (POP) as a separate recursor instance.

The second factor is *caching*. Cached responses bypass recursive resolution, potentially hiding the actual egress paths. We mitigate this by querying unique subdomains, effectively preventing cache hits and forcing upstream resolution.

The third and most important factor is the *egress structure* of the recursor. As summarized in Table 2, we distinguish four types. Recursors in R_1 only have a single fixed egress IP, making enumeration straightforward. R_2 and R_3 both expose multiple egresses and employ request-level load balancing: in R_2, load balancing is IP-based (e.g., hashing client's IP address), whereas in R_3, the selection follows less predictable strategies such as randomized dispatch. These egresses work independently, and each request is typically handled by exactly one of them. For R_3, complete enumeration of the egress set is not guaranteed. To handle this, we apply a statistical model to estimate the number of requests required for high-confidence coverage, as detailed in Appendix C. In contrast, R_4 recursors feature *collaborative* egress sets, in which request processing may involve multiple backend egresses, for example, via a shared backend infrastructure.

Table 2: Taxonomy (Recursive Resolution).

Egress			Notation	Anchor
Single			R_1	✓
Multiple	Independent	IP-based LB	R_2	✓
		Any Other LB	R_3	-
	Collaborative		R_4	✓

For each probe p_i and resolver $o_j \in O$, we perform t rounds of probing and record the observed egress set per round as $E_{ij} = \{E_{ij}^1, E_{ij}^2, ..., E_{ij}^t\}$. Each probe request is intentionally crafted so that the resolver must perform a full recursive resolution, typically by referencing a unique, deep subdomain, which is intended to bypass caches and trigger multiple outgoing queries.

3.2.2 Egress Anchoring

The ODNS infrastructure exhibits complex inter-resolver dependencies: many resolvers, rather than performing recursive resolution themselves, rely on upstream servers—often other resolvers—to complete queries. This layered dependency is opaque to external observers, making it challenging to attribute observed egress traffic to the actual entity responsible for resolution. To enable systematic analysis of such dependencies, we introduce the notion of an *Egress Anchor*.

Formally, we define an anchor as a one-to-many mapping

Algorithm 1: Egress Anchor Identification

Input:
$p_i \in P, o_j \in O, E_{ij} \in E_{obs}$

Initialize:
$F = \emptyset, A = \{E^{(x)} \mapsto \{(p,s)\}, D = \{(p,s) \mapsto \{E^{(x)}\}\}$

for *each* E_{ij} **do**
  if $\forall E^t_{ij} \in E_{ij},\ |E^t_{ij}| = 1$ **then**
    if $\forall E^p_{ij}, E^q_{ij} \in E_{ij},\ E^p_{ij} = E^q_{ij}$ **then**
      // $o_j \in R_1 \cup R_2 \vee o_j \in F, |U_j| = 1$
      $A[E^1_{ij}] = A[E^1_{ij}] \cup \{(p_i, o_j)\}$
    else
      // $o_j \in R_3 \vee o_j \in F, |U_j| > 1$
      $D[(p_i, o_j)] = D[(p_i, o_j)] \cup \{E_{ij}\}$
  else if $\exists E^t_{ij} \in E_{ij},\ |E^t_{ij}| = 1$ **then**
    // $o_j \in F, |U_j| > 1$
    $F = F \cup o_j$
  else if $\forall e^{t,n}_{ij} \in E^t_{ij},\ \exists! r_k \in R,\ e^{t,n}_{ij} \in E_k$ **then**
    if $\forall E^p_{ij}, E^q_{ij} \in E_{ij},\ \exists e^{p,m}_{ij} \in E^p_{ij},\ e^{q,n}_{ij} \in E^q_{ij},\ e^{p,m}_{ij} = e^{q,n}_{ij}$ **then**
      // $o_j \in R_4 \vee o_j \in F, |U_j| = 1$
      $A[\bigcup E^t_{ij}] = A[\bigcup E^t_{ij}] \cup \{(p_i, o_j)\}$
    else
      // $o_j \in F, |U_j| > 1$
      $F = F \cup \{o_j\}$
  else
    // $o_j \in F, |U_j| > 1$
    $F = F \cup \{o_j\}$
return F, A, D

from an egress set $E^{(x)} \subseteq E$ to the set of probe–resolver pairs (p_i, o_j) that consistently and exclusively trigger that set. We use ***ingress*** to refer to the specific $(clientIP, resolverIP)$ tuple through which a request reaches the resolver. An anchor, in contrast, captures the relationship between that ingress and the egress set. We record these stable relationships in a mapping A, where each entry is of the form:

$$A[E^{(x)}] = \{(p_i,o_j) \in P \times O \mid\ (p_i,o_j) \to E^{(x)} \\ \wedge\ \forall E^{(y)} \neq E^{(x)},\ (p_i,o_j) \nrightarrow E^{(y)}\}.$$

This construction allows us to treat each such (p_i, o_j) pair as a reliable "anchor" for the associated egress set $E^{(x)}$, facilitating later inference of dependency relationships between resolvers.

Importantly, the presence of an egress anchor does not necessarily identify the ultimate upstream recursor responsible for resolution. Rather, it supports only a behavioral equivalence class. For instance, the observed behavior is consistent with either (i) $o_j \in R_1 \cup R_2$, or (ii) $o_j \in F$ with $|U_j| = 1$. These two cases are *indistinguishable under our current measurements*, as both result in the same stable egress pattern. We treat them equivalently during anchor-based reasoning and describe a dependency ordering method in Appendix B, though it is not compulsory for subsequent analysis.

However, not all (p_i, o_j) pairs exhibit this stable behavior. Some may trigger different egress sets across measurement rounds or respond inconsistently due to multi-upstream forwarding. We record such cases in a separate mapping D, which maps each ambiguous pair to the full set of egress sets it triggers:

$$D[(p_i,o_j)] = \left\{E^{(x)} \subseteq E \mid (p_i,o_j) \to E^{(x)}\right\}.$$

Finally, we maintain a set F of resolvers that are conclusively identified as *forwarders* based on specific observable patterns. Once such a classification is established, no further anchor-based analysis is necessary for these resolvers, as their role within the ODNS hierarchy has already been determined. These resolvers are thus excluded from subsequent anchor inference but remain relevant in the broader dependency structure.

The process for assigning anchors, detecting ambiguous mappings, and identifying confirmed forwarders is formalized in Algorithm 1, which examines observed egress sets and populates A, D, and F accordingly.

As defined in Section 3.1, forwarders may perform *concurrent forwarding*, where a request is forwarded to multiple upstream resolvers simultaneously. Later in Section 4.3, we demonstrate how multi-egress observations arising from such behavior can be either confidently attributed to a recursor or disambiguated across upstreams.

3.2.3 Egress Grouping

A central challenge in identifying recursors for self-amplification analysis is that many resolvers do not expose a stable mapping between an ingress and any individual egress. For resolvers that employ randomized or unpredictable dispatching strategies, repeated requests received from the same ingress may reach different egresses across measurement rounds. This instability prevents us from treating each egress as an independent amplifier and makes it impossible to attribute amplification behavior to a specific backend component. Reliable estimation of amplification thresholds requires an amplifier that can be consistently exercised; otherwise, the observed MAF becomes non-reproducible and cannot be linked to the resolver's actual recursion logic. The ultimate goal is to create a stable abstraction that enables systematic and comparable amplification measurements across diverse resolver architectures.

This motivates the need to identify the smallest group of egresses that can be triggered in a reproducible manner and used as the basis for amplification measurement, which we refer to as the *minimal set*.

This unit varies depending on the type of recursor. For R_1 resolvers, which expose a single, fixed egress IP across all

measurement rounds, the minimal set is simply the egress IP itself. In the case of R_2 resolvers, which utilize IP-based load balancing among multiple egresses that operate independently, each individual egress can be deterministically and stably triggered from specific probes. As such, each egress IP constitutes its own independent minimal set.

In contrast, R_3 resolvers employ independent egresses but use unpredictable dispatching strategies. As a result, individual egresses cannot be stably isolated, and we treat the entire set of observed egresses associated with an R_3 resolver as a single collective minimal set. However, from an observational perspective, this behavior is indistinguishable from multi-upstream forwarding (e.g., f_2 in Figure 1), as repeated probes from the same ingress may trigger different egress sets across measurement rounds. Similarly, R_4 resolvers adopt a collaborative scheduling model, wherein multiple egresses may jointly respond to a single request. In such cases, we consider the union of all egresses observed across measurement rounds as a single minimal set.

Based on this classification, only R_3 resolvers require additional handling to resolve ambiguous cases. To this end, we rely on the mapping D defined earlier. These entries correspond to (p_i, o_j) pairs that trigger multiple egress sets across rounds. To separate these cases, we first aggregate all egress sets associated with the same resolver o_j:

$$\mathcal{D}_j = \bigcup D[(p_i, o_j)] \quad \text{for all fixed } o_j.$$

We then apply the following rule: for any pair of resolvers o_j and o_k, if $\mathcal{D}_j \subsetneq \mathcal{D}_k$, we classify o_k as a *forwarder* and add it to the set F, since its egress behavior is fully subsumed by that of o_j thus not suitable for the subsequent SAA analysis. Intuitively, this subset relation implies that o_k inherits all egresses of o_j and introduces additional egresses, which is characteristic of upstream forwarding rather than independent recursion. Although this algorithm has quadratic worst-case complexity, it is applied only to a small number of resolvers, making the cost negligible in practice.

The remaining resolvers form the *minimal sets*, representing recursors that possess multiple independent egresses and do not rely on upstream resolution. We then group all associated (p_i, o_j) pairs as their ingresses for reliable measurement of these egress sets.

3.3 Maximizing Amplification Potential

The impact of self-amplification attacks on real-world DNS infrastructure remains uncharted. While analyzing individual vulnerabilities in isolation is already an onerous task, reasoning about real attacks must further deal with a diversity of intricate aspects. To start with, we systematize and generalize the adversary models considered by existing self-amplification attacks.

3.3.1 Adversary Models

We consider an adversary whose goal is to induce excessive outgoing queries at selected resolvers by exploiting self-amplification vulnerabilities. To mount such attacks, the adversary operates authoritative nameservers under its control and installs specially crafted DNS zones that embody resolution configurations that can trigger pathological query behavior.

The adversary controls a set of hosts, referred to as bots, that can issue DNS requests to chosen resolvers. These bots must have permission to use the target resolvers for recursion. We assume the adversary does not require on-path capabilities, spoofing, or control over the recursive resolvers themselves. Each bot can send DNS requests constructed to invoke specific amplification primitives deployed on the adversary's nameservers. And the adversary is free to deploy many independent instances (i.e., different QNAMEs that trigger the same primitive) of the most effective resolution configurations. Zone content may be provided statically through standard zone files or generated dynamically in response to correlated queries. Within these capabilities, the adversary attempts to maximize the outgoing queries (i.e., MAF) induced per request.

3.3.2 Resolver-Specific Composition

The amplification capacity of a DNS resolver is governed not only by the structure of the resolution configurations, but also by its internal threshold constraints on individual resolution primitives. To construct the most effective test case for each open resolver, we follow a two-phase strategy inspired by CAMP [10] that combines threshold discovery with compositional amplification.

To identify how aggressively each open resolver supports a given amplification primitive, we apply a *bounded binary search* approach. For each primitive, we define a safe and configurable lower and upper bound (e.g., number of rewrite $\in [1, 50]$, aligned with the default `max-recursion-queries` setting in BIND9) and probe within this range using binary search. In each iteration, we craft a resolution configuration using the target primitive at a specific parameter value and send a request to the resolver. Based on whether the request leads to additional outgoing queries, we adjust the search bounds. This process minimizes the number of queries that each resolver generates. By testing only a small set of parameter values rather than every value up to the threshold, it avoids repeatedly triggering large numbers of outgoing queries, which is essential for ethical large-scale measurement.

After determining the usable range of each primitive, we construct high-MAF test cases by composing them in two dimensions: *breadth* (e.g., rewrite chain) and *depth* (e.g., dense delegation). Starting from the most potent primitives, we incrementally assemble resolution configurations tailored to each resolver. At each step, we send a request to the resolver and monitor its outgoing queries to compute the MAF of the composition. Patterns that result in resolution suppression

(e.g., due to loop detection) are pruned from further composition.

The result is a resolver-specific resolution structure that maximizes observable amplification, while respecting each resolver's individual operational limits.

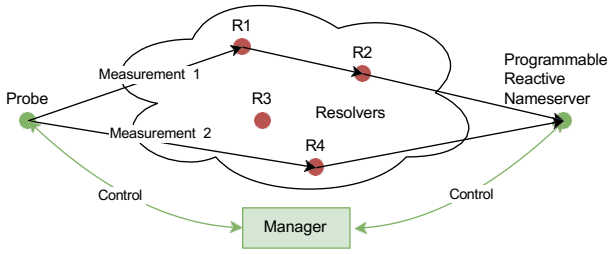


Figure 2: Overview of DaLens where a manager orchestrates measurements across probes, programmable reactive name server, and resolvers.

4 DaLens Framework

The methodology in Section 3 establishes the overall approach for characterizing self-amplification behavior, but executing it at Internet scale requires a flexible and coordinated system. In practice, we must reliably bypass caching and forwarding interference, construct diverse resolution configurations on demand, track resolver behavior across recursive sessions, and automate measurements across globally distributed vantage points. No existing DNS measurement toolchain provides these capabilities in a unified and scalable manner.

To address this challenge, we design and implement DaLens, a measurement framework that provides the programmable control, stateful responsiveness, and large-scale automation necessary to execute our methodology in practice.

4.1 Design Principles

DaLens is guided by four design principles that capture the fundamental requirements for measuring recursive resolver behavior in a controlled and systematic manner, with a particular focus on SAA vulnerabilities within the ODNS infrastructure.

Programmability. SAA measurements demand arbitrary and non-standard resolution behaviors, such as controlled rewrite, referral, and delegation sequences. DaLens treats resolution logic as programmable rather than zone-file–driven, using configuration-defined response generation to construct precisely controlled resolution processes.

Statefulness. Analyzing recursive behavior requires tracking how individual sessions progress, including detecting concurrent forwarding or suppression. DaLens embeds per-session identifiers into QNAMEs and maintains lightweight state, enabling differentiated responses to identical queries and fine-grained tracing of recursive execution.

Automation. Internet-scale measurements must be executed reproducibly across many probes and configurations. DaLens incorporates automated orchestration for configuration management, request generation, log collection, and analysis execution, ensuring consistent and high-throughput experimentation.

Together, these principles shape DaLens as a comprehensive measurement system: programmability and statefulness provide control and visibility at the authoritative interface, while scalability and automation support systematic execution of the methodology at Internet scale. We emphasize that these capabilities are not available in conventional nameserver architectures, making DaLens indispensable for rigorous, comprehensive analysis of recursive resolvers across diverse ODNS measurement settings.

4.2 System Components

Figure 2 illustrates the high-level architecture of DaLens, which consists of three main components: (1) a *programmable reactive nameserver (PRN)*, (2) *probes*, and (3) a *manager*.

4.2.1 Programmable Reactive Nameserver

The PRN is the core of the system. It introduces two key abstractions: *resolution configuration* and *resolution instance*.

A resolution configuration defines a response-generation template that specifies an ordered sequence of DNS responses to be returned to a resolver, and is identified by a unique *configuration ID (conf ID)*. A resolution instance represents a single execution of a configuration and is identified by a unique *instance ID*, allowing multiple independent resolution sessions to follow the same configuration in parallel. The pair (conf ID, instance ID) therefore uniquely identifies a resolution context.

We encode ⟨conf ID, instance ID⟩ into the QNAME, specifically within a dedicated subdomain of the PRN-controlled domain (e.g., `a.conf-x-ins-y.[domain]`, where `x` denotes the configuration ID and `y` the instance ID). This placement is deliberate: even when QNAME minimization (QMIN) is enabled, this subdomain is preserved across all queries within a resolution context, ensuring that the PRN consistently receives the same identifiers throughout the resolution process.

When receiving such a query, the PRN uses the configuration ID to locate the selected resolution configuration and the instance ID to identify the corresponding resolution instance. Each instance maintains an internal counter that tracks its progress through the configuration's response-generation template. After sending a response, the PRN increments the counter, such that the next query with the same (conf ID, instance ID) triggers generation of the subsequent response in the template.

4.2.2 Probes and Manager

Probes are stateless packet generators that transmit DNS queries from geographically distributed network locations. Each probe executes probing tasks issued by the manager and does not maintain resolution, session, or experiment state. Probes are solely responsible for emitting queries from diverse ingress points to trigger resolver behavior.

The manager defines experiment semantics and coordinates system execution. It assigns a globally unique identifier to each client request and allocates instance IDs for resolution instances, enabling unambiguous association among queries, ingress points, and resolution contexts. The manager injects these identifiers into queries issued by probes, send resolution configurations to the PRN, and schedules probing tasks across probes. It maintains global experiment state and correlates logs collected from probes and the PRN to reconstruct resolver behavior at scale.

This architecture allows DaLens to scale beyond small test cases: experimenters can deploy hundreds of configurations and sessions without manual intervention, while maintaining reproducibility and consistency across runs.

ins	prog	egress_ip	qname	qtype
1	1	138.68.96.43	a.**conf**-x-**ins**-x	A
1	2	138.68.96.43	a.**conf**-x-**ins**-x	A
3	1	159.203.105.218	d1.**conf**-x-**ins**-x	NS
4	1	159.203.105.218	r1.**conf**-x-**ins**-x	A
5	1	146.190.91.208	c1.**conf**-x-**ins**-x	A

Figure 3: PRN Log Format.

4.3 Case Studies

To demonstrate how DaLens can be applied in practice, we present two representative case studies, and the corresponding resolution configurations are shown in Appendix E.

Recursor Identification. Recursor identification relies on correlating ingress and egress points. Probes send requests with unique instance IDs for each ingress point, and the PRN records these IDs along with the corresponding egress IPs. Since each QNAME with a unique instance ID is queried only once, this method bypasses caching and reliably triggers resolution. This allows us to derive the egress set associated with each ingress point.

To handle concurrent forwarding, we configure a unified resolution configuration containing multiple CNAME records, which redirect resolvers to different configurations with distinct IDs. For example, a request `a.conf-1-ins-1.[domain]` may be forwarded to multiple resolvers that perform resolution concurrently. In `conf-1`, we define two CNAME templates: one pointing to `conf-2` and another to `conf-3`. When the first resolver queries `conf-1-ins-1`, it is redirected to `conf-2-ins-1.[domain]`. After the counter for `ins-1` in `conf-1` is incremented, a subsequent resolver querying `conf-1-ins-1` instead receives a CNAME to `conf-3-ins-1`. This allows us to identify concurrent forwarding from PRN logs (see Figure 3).

MAF Measurement. Once we have the target resolvers, we can leverage DaLens to systematically measure their MAF. We configure a diverse set of resolution configurations that exercise both primitive and compositional amplification patterns, and instruct probes to issue requests accordingly. Each request embeds a unique instance ID to facilitate matching between the probe-side request and server-side activity.

By combining the probe's request metadata with logs collected at the PRN, we derive the threshold of each resolver using two main criteria. For fanout, since resolvers may randomly select or retry different NS records, we compute MAF by counting the number of distinct queries observed across candidate NS targets. In contrast, for referral and rewrite chains that require sequential resolution, the traversal depth can be inferred by locating the maximum template ID reached within a given resolution session, as each stage is executed in order. For dense delegation, resolvers may not enable QNAME minimization and instead query the full domain, making it impossible to infer depth from QNAME labels alone. In such cases, we adopt a counting-based approach, using the number of unique queries sent as the threshold estimate. This strategy is similarly applied to derive the MAF of compositional patterns that combine multiple primitives.

5 Measurement Results

We have applied DaLens to conduct a series of measurements on open resolvers. All the resolution configurations involved are specified using our own domains hosted by our nameservers. We employ six probes in Europe, North America, Oceania, and Asia for the measurements.

Our measurement pipeline consists of four phases. First, we identify open resolvers in the IPv4 space and exclude non-functional or irrelevant ones. Second, we apply our identification method to identify the recursors in the ODNS infrastructure and its downstream forwarders. Third, we probe each recursor to determine its thresholds for individual amplification primitives. Finally, we compose these primitives to measure worst-case MAFs.

We further provide a quantitative analysis of the resolution workload imposed on resolvers in Appendix D, demonstrating that our measurement adheres to ethical guidelines.

5.1 Discovery and Preprocessing

Our focus so far has been on open resolvers capable of resolving most domains globally and accessible from all probes. We say *most domains* because there are resolvers implementing filtering policies for censorship or security reasons.

Table 3: Summary of Open Resolver Discovery

Round	Date	Probes	# Resolvers
1	March, 2025	FRA, NYC, SYD, SGP	2,082,889
2	July, 2025	FRA, NYC, SYD, SGP	1,835,277
3	August, 2025	AMS†, SFO†, SYD, SGP	1,647,158

† In the latest round, we replaced two probes as a precaution to diversify vantage points and mitigate potential measurement interference.

We use XMap [19] to send A-record requests to all public IPv4 addresses. Each IP receives two pseudorandom requests over UDP and two over TCP to avoid false negatives caused by network issues. We classify a host as a resolver if it responded to at least one request. Our multiple rounds of discovery indicate that the universe of open resolvers is highly dynamic, a phenomenon also noted in previous work [43]. Table 3 summarizes our three discovery rounds.

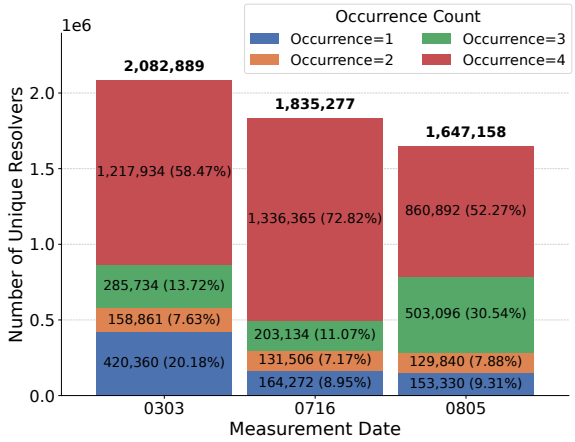


Figure 4: Illustration of observability.

Resolver Observability. Across all three rounds, we discovered 3,421,931 distinct resolvers. However, their intersection contains only **780,430** resolvers, suggesting a significant churn rate. In practice, an attacker is likely to favor resolvers that persist over time, because such resolvers are intentionally operated as open services, serving a broader client population and allowing an attacker to cause more severe damage if they are rendered unavailable. It is also unlikely for nameservers to block persistent resolvers, as doing so may cause severe collateral damage.

Figure 4 further breaks down the resolvers discovered in each round by their *occurrence count*, i.e., the number of probes from which a resolver was observed to respond. This partial visibility may be due to factors such as IP-based filtering, geographic restrictions, or transient reachability issues.

For subsequent analysis, we restrict our attention to resolvers with an occurrence count of 4, i.e., those observable from all probes within the same round. We argue that resolvers with broader public accessibility are more likely to matter from an SAA perspective, because short-lived resolvers typically have very few downstream clients and, even if they technically allow high MAF, exploiting them would yield limited impact.

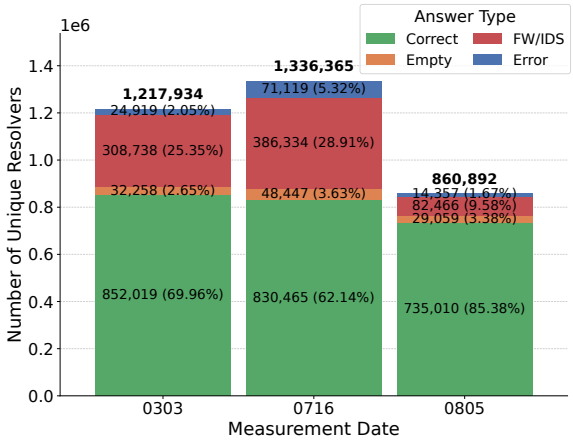


Figure 5: Illustration of correctness.

Answer Correctness. Next, we evaluate the correctness of answers returned by the resolvers observable from all probes in each round. Figure 5 categorizes the responses into four groups: *Correct* answers matching the ground truth, *Empty* answers where no records were returned, responses intercepted by *Firewall/IDS* systems, and *Error* answers where the returned records are incorrect.

A substantial fraction of the observed servers fell into the *Firewall/IDS* category. Upon closer inspection, we found that many of these servers responded with a `CNAME` record redirecting our request to a specific sinkhole domain (e.g., `sinkhole.paloaltonetworks.com`). However, when we subsequently queried these servers for other popular domain names, they ignored the requests entirely and provided no responses.

We aggregated the IP addresses of these "resolvers" by their Autonomous Systems (ASes) using the IPinfo Lite database [1] and observed that they were concentrated in thousands of ASes, primarily within academic institutions and the financial sector. This suggests that these entities are not genuine resolvers but rather end hosts protected by firewall or intrusion detection systems, or in some cases, unused IP addresses. Our measurement domains were likely flagged as threat intelligence entries propagated across deployments of the same security products, leading to the consistent sinkhole behavior.

We filter our dataset to contain resolvers that returned correct answers in all three measurement rounds, yielding a total of **307,397** resolvers. We regard these resolvers as the genuine public resolvers, as those returning empty or erroneous answers are likely misconfigured, non-recursive, or otherwise unsuitable for ODNS measurement. Note that even these resolvers are not always responsive, and we indicate the number

of resolvers responding to our requests for each measurement.

Addressing Method. Before proceeding to our main measurements, the final preprocessing step is to identify resolvers that employ *anycast* addressing.

We treat these resolvers separately for two reasons. First, because anycast infrastructures typically operate multiple geographically distributed points of presence (PoPs), a larger number of probes located in different networks is required to trigger them comprehensively. Second, such resolvers are more likely to be *recursors* rather than forwarders, as anycast deployment is operationally advantageous for large-scale recursive services: it improves fault tolerance, load balancing, and user proximity, while the complexity of coordinating multiple PoPs generally outweighs the operational simplicity needed for forwarders.

We obtain the list of anycast IPs from the same database [1] and identify a total of **7,764** anycast-based recursors in our dataset. We then use about 1,000 globally distributed RIPE Atlas probes, each sending requests with unique instance IDs, to identify their individual PoPs. By comparing the resulting egress sets, we infer the number of PoPs and determine which probes reach which PoP. Each identified PoP is then treated as an independent recursor instance. Anycast recursors identified in this manner do not require egress anchor identification and are directly measured to assess their configuration and amplification power.

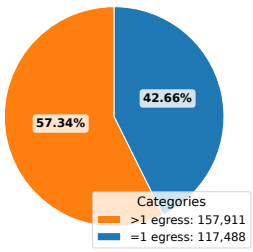


Figure 6: Resolver Set (Total = 276,399).

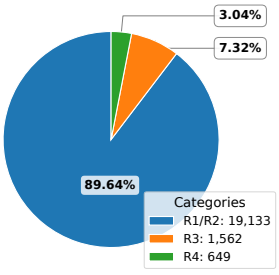
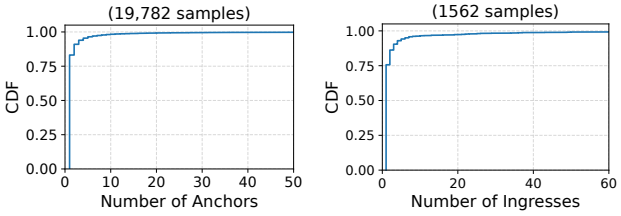


Figure 7: Category Distribution (Total = 21,344).

5.2 ODNS Infrastructure

We now apply our recursor identification method to the set of ODNS resolvers discovered in the previous section. We obtain a final set of 276,399 unique resolvers that respond to our further probes and for which egress enumeration and attribution can be reliably performed.

As shown in Figure 6, egress multiplicity is a widespread phenomenon: more than half of these resolvers trigger multiple egresses during a single resolution process. In the following subsections, we apply our recursor identification pipeline to characterize these resolvers, derive their structural roles, and ultimately build a measurement-grounded view of the ODNS infrastructure.



(a) $R_1/R_2/R_4$ Egress Anchor (b) R_3 Ingress

Figure 8: Egress Anchor Distribution

Recursor Type Distribution. Using the methodology described in Section 3.2, we classify each resolver in our dataset into one of four recursor types (R_1–R_4) based on its observed egress behavior. These types correspond to distinct resolution models introduced earlier and determine how reliably we can measure each resolver's amplification potential.

Figure 7 shows the distribution across all identified recursors, totaling 21,344 unique egress sets. This represents **6.94%** of the 307,397 resolvers, a proportion consistent with prior work [39], which also observed that the majority of open resolvers function as forwarders rather than full recursors.

Among the identified recursors, types R_1 and R_2 dominate. These exhibit deterministic single-egress behavior and are more likely to be operated by individuals or small organizations with limited infrastructure needs. In contrast, R_3 and R_4 exhibit more complex scheduling patterns, involving either randomized dispatch or collaborative processing. These require more sophisticated setups and are often maintained by large-scale operators or service providers. Although they are smaller in number, their scale and structure make them particularly attractive—and vulnerable—targets for DoS attacks, especially under self-amplification scenarios where multi-egress behavior can inflate total workload.

Key Observation 1: ODNS is structurally fragile–an attacker only needs to target a few.

Out of 307K responsive public resolvers, only 9.5% (29K) are true recursors. This imbalance exposes a small, critical core whose compromise can affect vast portions of the DNS ecosystem.

Egress Anchor. According to our identification results, resolvers of type R_1, R_2, and R_4 are associated with stable *egress anchors*–probe-resolver pairs (p_i, o_j) that consistently trigger a single egress set. By definition, each anchor's upstream is uniquely determined: the resolver forwards queries exclusively to one specific egress set, with no observable fallback or diversity.

This structural property has significant implications for amplification: any vulnerability in the anchored egress set

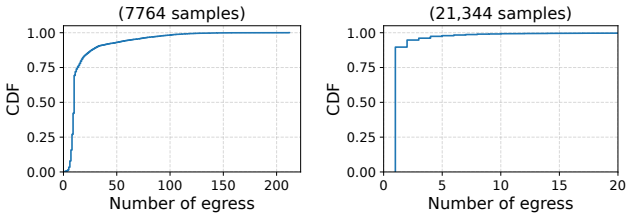
directly compromises the entire downstream resolver, with no redundancy to mitigate the attack. Figure 8a shows the distribution of anchor counts per egress set. We observe that a large fraction of egress sets have only a few associated anchors, indicating a tightly coupled forwarding relationship.

This tight binding further implies that the anchors are largely interchangeable from a measurement perspective. Therefore, in the subsequent analysis of SAA-related factors, we can reliably measure a given egress set by randomly selecting any of its anchors.

For resolvers in R_3, due to their randomized dispatching across independent egresses, no probe-resolver pair can consistently trigger any specific egress. As a result, we group all egresses of an R_3 resolver into a single unit, which we refer to as its *ingress* (Figure 8b). Although individual egresses cannot be isolated for probing, this ingress abstraction allows us to treat R_3 resolvers consistently with the others regarding dependency analysis.

Key Observation 2: Grouping reveals ingress triggers for deterministic amplifiers.

By grouping egresses into 29K minimal sets, we uncover 110,129 forwarders that deterministically dispatch to them—exposing stable ingress paths attackers can exploit. The remaining 189,504 forwarders use multiple upstreams, making them less attractive for targeted SAA.



(a) Anycast Recursor (b) Unicast Recursor

Figure 9: Egress Set Size Distribution

Egress Set Size. We next examine the size of each recursor's egress set—i.e., the number of distinct IPs used for outbound queries during recursive resolution. This metric offers a coarse but useful indicator of the resolver's backend architecture, fault tolerance, and potential resilience to attack.

Figure 9 shows the cumulative distribution of egress set sizes for both anycast and unicast recursors. We observe that anycast deployments typically exhibit significantly larger egress sets, with some using over 200 distinct egress IPs. This reflects the inherently distributed nature of anycast architectures, where requests may be served by multiple physical instances. In contrast, unicast recursors tend to have much smaller egress sets: over 90% of them expose fewer than 5 egresses, with the median being just one. This aligns with our earlier observation that many open resolvers operate as simple forwarders or are deployed by smaller organizations.

From the perspective of SAA vulnerability, large egress sets imply greater redundancy and operational tolerance. Specifically, even if one backend is vulnerable or degraded, others may continue serving queries, reducing the amplification effect. Conversely, recursors with smaller egress sets are more brittle: targeting a single vulnerable egress may be sufficient to amplify traffic for the entire resolver.

Concurrent Forwarding. A final architectural trait we examine is the use of *concurrent forwarding*, where a resolver forwards the same client request to multiple upstream resolvers in parallel. Among the resolvers in our dataset, we identify **7,532** that exhibit this behavior based on observed query duplication patterns.

Concurrent forwarding is typically adopted as a performance optimization, allowing the resolver to return the first successful response while discarding the rest. While beneficial for latency, this behavior significantly increases the total number of outbound queries per client request, which in turn amplifies the potential impact of self-amplification attacks. In the worst case, if each of the upstreams contacted by the forwarder is itself vulnerable to amplification, the total magnification effect compounds—one client request may result in multiple redundant resolution cascades. Hence, concurrent forwarding acts as another *multiplier* for MAF, and its presence must be carefully accounted for.

5.3 Recursor Configurations

In this subsection, we discuss results for factors that dictate an individual resolver's behavior based on the identified 21,344 unicast and 7,764 anycast recursors. For anycast recursors, the reported results are obtained by averaging the measurement results across their corresponding PoPs.

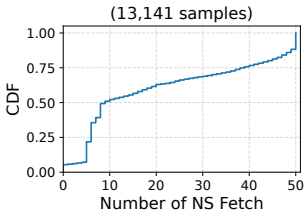


Figure 10: Fanout Distr.

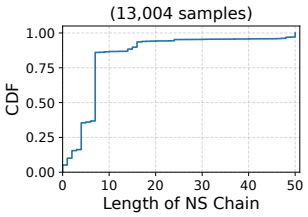


Figure 11: Referral Distr.

NS Fetch. For the NS Fetch pattern, we process the measurement results to obtain the number of NS records fetched. Figure 10 reports the results.

We observe substantial variation in resolver behavior: nearly **50%** of resolvers fetch more than 10 NS records, and a non-negligible fraction exceeds 30. This indicates that resolvers are willing to perform extensive upstream lookups before giving up on unresolvable NS names.

Such aggressive behavior bodes ill in the context of self-amplification attacks, where NS records are deliberately constructed to be unresolvable. Attackers can exploit this leniency to maximize outbound queries per resolution attempt. The results here establish a lower bound, as our pattern contains at most 50 NS records—resolvers with higher internal limits may fetch even more.

NS Chain. For the NS Chain pattern, we measure how deep resolvers are willing to follow referral chains before terminating resolution. Figure 11 summarizes the results across 13,004 resolvers.

We find that around **15%** of resolvers accept referral chains longer than 10, and roughly **8%** follow more than 20 referrals. While most resolvers apply some upper bound, a small but significant fraction tolerates chains of length close to 50—the limit imposed by our test pattern. Such leniency has direct consequences for self-amplification attacks, where referral chains are artificially extended to force iterative lookups and inflate outbound traffic. The high tolerance observed here suggests that attackers can reliably extract large amplification with minimal effort, simply by chaining NS delegations.

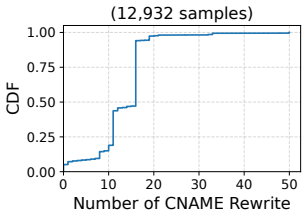


Figure 12: Rewrite Distr.

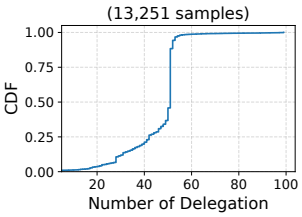


Figure 13: DDLG Distr.

CNAME Chain. We measured how long resolvers are willing to follow CNAME chains, another potent amplification vector. Figure 12 shows the results across 12,932 resolvers, using test patterns with a chain length of 50.

We observe that roughly **6%** of resolvers terminate after a single rewrite, whereas over **85%** follow at least 10 CNAME redirections. Notably, some resolvers traverse the entire chain, hitting the upper bound defined in our test. The long tail in the CDF indicates that a non-negligible number of resolvers tolerate extremely deep CNAME chains, significantly amplifying query workloads in adversarial settings.

Dense Delegation. We conclude our primitive analysis with the *dense delegation (DDLG)* pattern. Unlike patterns that exploit name rewriting or aliasing, dense delegation constructs QNAMEs where *a distinct zone authority backs each label*. This forces resolvers to follow the delegation chain one level at a time, issuing a new query at each step to discover the next authoritative server.

As shown in Figure 13, resolvers exhibit a wide spectrum of tolerance for delegation depth. While most resolvers can handle over 40 delegations, around 1% proceed through more than 60 distinct authorities, and a small subset reach close to the protocol-defined maximum.

Compared to CNAME chains or NS fanout, dense delegation offers a lower per-hop amplification, but guarantees a predictable cost for each delegation step. Its deterministic structure and compatibility with standard resolution logic make it especially attractive for crafting test cases that maximize amplification while remaining within protocol bounds.

Key Observation 3: Resolvers are structurally lenient, and amplification opportunities abound.

Despite structural differences across patterns, most resolvers exhibit high tolerance for deep or wide resolution structures, creating ample opportunities for predictable and sustained amplification.

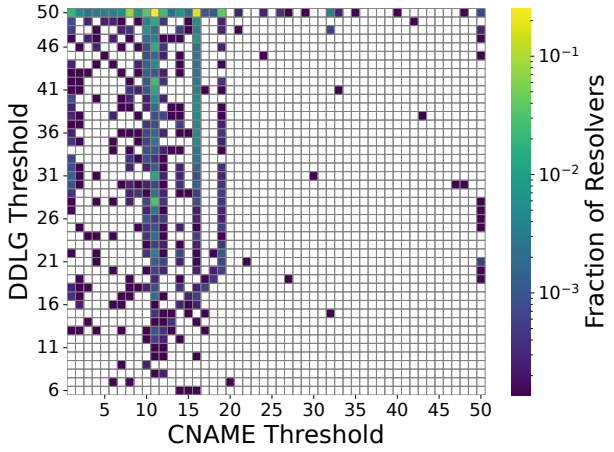


Figure 14: Distribution of MAF versus resolution time.

5.4 Estimating Amplifier Power

The limits for individual features reported in the previous subsection are far from capturing the full amplification potential of resolvers. We push this further by measuring several amplification profiles of compositional nature as suggested by a recent study [10].

To better understand the amplification potential of resolvers under composition, we jointly examine two of the most impactful resolution behaviors: CNAME rewrite and dense delegation.

Figure 14 plots the joint distribution of CNAME rewrite threshold and dense delegation threshold across all tested resolvers. Each cell represents the fraction of resolvers tolerating the corresponding combination of limits, in log scale. We observe that rewrite thresholds tend to be more concentrated. In contrast, DDLG thresholds are more broadly distributed, ranging from 10 to nearly 50, suggesting greater diversity in how resolvers handle recursive delegation.

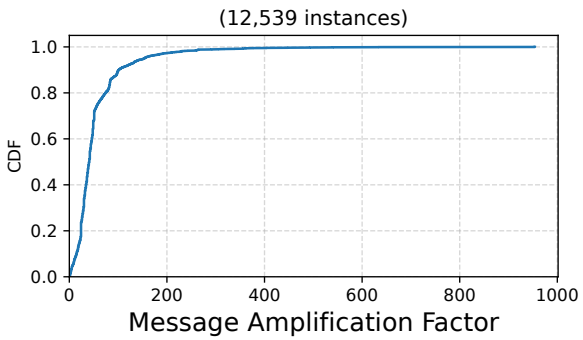


Figure 15: Resolver Distribution by CNAME vs DDLG Threshold (Log Scale).

This diversity matters. If a resolver supports high thresholds for both primitives, it can incur substantial work even for modest requests. Yet interestingly, few resolvers fall into the top-right corner of the matrix: the population that supports both long rewrite and deep delegation is limited. This suggests resolver developers may impose global safeguards or stop conditions that transcend the limits of individual primitives. We further quantify this by examining actual amplification behavior. As shown in Figure 15, we measure the MAF for 12,539 resolvers using carefully composed query patterns.

The results reveal a clear long-tail distribution: while 90% of resolvers cap their MAF below 100, around 2% of resolvers exceed a factor of 200, and the most aggressive ones generate up to nearly 1,000 queries for a single resolution. These high-MAF resolvers are of particular concern in practice, as they pose substantial DoS risk even with limited adversarial bandwidth. Taken together, Figure 14 and Figure 15 illustrate that amplification composition is not trivially multiplicative. That is, the maximum MAF is not simply the product of rewrite and delegation thresholds. This implies that many resolvers internally enforce additional global constraints (e.g., per-query caps, recursion budget, cycle detectors), which act as implicit limits on the actual work performed.

Key Observation 4: Composed behaviors silently unlock extreme amplification.

Resolvers with moderate individual thresholds can exhibit severe amplification when factors are composed, reaching MAFs near 1,000. This enables highly asymmetric attacks where minimal effort yields disproportionate disruption.

The results above suffice to demonstrate the great amplification potential of open resolvers. Despite this, our measurements are still *underestimates*. The maximum MAF achievable by the resolvers can be much higher. Finding out such limits per se poses challenges and would require more intrusive probing, which raises ethical concerns; we leave this to future work.

5.5 Implications for SAA Vulnerabilities

To connect our measurements with the security implications of self-amplification vulnerabilities, we interpret the results through the lens of how readily an adversary can exploit the observed resolver behavior. The recursor identification pipeline reveals a small but structurally critical subset of resolvers on which a significant fraction of forwarding resolvers depend. These recursors constitute the effective amplification surface that an attacker can target, and their correct identification is essential for understanding the practical reach of self-amplification attacks.

The primitive thresholds we measure further indicate how permissive these recursors are with respect to the mechanisms that underpin known self-amplification vulnerabilities, including deep referral chains, repeated NS fetching, and iterative alias rewrites. High tolerance for these behaviors directly increases the amount of recursive work that a resolver is willing to perform on behalf of an untrusted client, thereby expanding the adversarial leverage available to cause damage.

Our composed amplification measurements then quantify the combined effect of these behaviors and reveal that even resolvers with moderate individual limits can produce substantial message expansion when primitives interact. The resulting long-tail distribution, with some resolvers generating nearly one thousand recursive queries per request, demonstrates that these vulnerabilities remain practically exploitable in real-world deployments.

This characterization also enables meaningful *triage*: by identifying which resolvers contribute most significantly to the amplification surface, DNS operators can prioritize mitigation efforts toward the infrastructures whose compromise would result in disproportionate impact.

Taken together, these observations show that the vulnerabilities documented in prior work remain prevalent across diverse resolver architectures, providing empirical evidence that today's ODNS ecosystem is structurally susceptible to self-amplification attacks.

6 Disclosure

We have directly disclosed the issues to DNS vendors, including BIND, Unbound, PowerDNS, and Knot, as well as participants in a recent DNS-OARC forum. Our main collaborator for coordinated disclosure, the National Cyber Security Centre (NCSC) of Switzerland, has issued a security advisory (TLP:AMBER+STRICT) to their partner GovCERTs and through the International Watch and Warning Network (IWWN). Moreover, the center has directly notified DNS operators (including a major global open resolver) in our country of residence.

In response, the above-mentioned (and other) DNS vendors have applied security patches to their software, and we have observed reduced amplification factors on some open

resolvers since our initial measurements. In addition, following a recent DNS-OARC event, an IETF draft has been put online to propose mitigation for amplification vulnerabilities. We plan to issue another security advisory to better inform and prepare DNS operators to defend against SAAs. We are continuing these efforts as part of the ongoing coordinated disclosure process.

7 Related Work

Measuring ODNS Infrastructure. While prior work has characterized various aspects of ODNS behavior and risk, none has provided a formalization of the ODNS infrastructure or its resolution structure. Several studies have investigated ODNS infrastructure in terms of prevalence, behavior, and security risks. Kührer et al. [18] and Park et al. [31, 32] showed that many open resolvers return incorrect or malicious responses and often deviate from standard behavior. Luo et al. [22] highlighted the dependency concentration in recursive resolution, finding that over 90% of ODNS rely on fewer than 9% of egress resolvers, many operated by third parties. Xu et al. [40] further revealed that over 90% of forwarding resolvers are backed by a small set of upstream resolvers, exposing systemic centralization. Liu et al. [20] examined DNS interception, showing widespread spoofing by on-path devices, especially toward public DNS services. Nawrocki et al. [29] identified transparent forwarders as a substantial and previously overlooked component of ODNS, with strong regional concentrations and heavy reliance on a small set of public resolvers. Other works [33,36] studied resolver caching and client-side DNS behavior.

Our work builds on these efforts by structurally analyzing ODNS resolution chains—specifically, identifying egress sets, anchors, and scheduling behavior using active measurements.

Measuring DNS-related DoS. Reflective amplification attacks (RAA) have been a subject of measurement research for years. Rossow [34] manually analyzed the vulnerabilities of popular UDP protocols, including DNS. AmpMap [25] makes a step towards automatically identifying vulnerable servers from packet header fields. Yazdani et al. [43, 44] examined the amplification effects of different types of DNS requests and discovered thousands of echoing resolvers that reply excessively to a single client request and thus increase the risk of RAA. In contrast to RAA, self-amplification attacks exploit sophisticated features inherent in DNS protocols, and therefore, they involve more factors to be analyzed and measured.

Moura et al. [28] evaluated, in retrospect, the effectiveness and dynamics of anycast deployments in defending DNS root nameservers against DDoS attacks. Sommese et al. [37] found that millions of domains experienced at least one DoS attack during a period of 17 months, some of them suffering severe performance degradation. These studies provide strong evidence for the DoS threats faced by today's DNS infrastructure. With the advent of self-amplification attacks, the need to thoroughly understand and quantify these threats through proactive (and periodical) measurements becomes only more pressing.

Measuring DNS Properties. Some of the DoS factors we consider in DaLens are also studied by prior work, with each study focusing on a single primitive rather than a compositional or systematic manner. Vries et al. [9] measured the adoption of QMIN by resolvers and their specific algorithms. Later, Magnusson et al. [23] report a significant increase in the adoption but additional subtleties in resolver behavior. Based on extensive empirical analysis, Moura et al. [27] provide guidelines on configuring proper TTL values for different situations. Bhowmick et al. [6] discovered unexpected TTL extension by resolvers that may impact dependent systems such as CDN. Randall et al. [33] unveiled the internal caching architecture of three large resolver systems through delicate measurement techniques. Prior work has measured specific resolver features that lead to amplification, including the CNAME chain length for the Unchained attack [8], the number of concurrent NS fetches for the NXNS attack [2], delegation loops for the TsuNAME attack [26], and the number of failover retires for the TsuKing attack [41]. In contrast, DaLens offers a generic framework with an expressive abstraction that captures almost any resolver-centric measurements and supports a variety of factors in an extensible manner.

PolarDNS [30] is an authoritative DNS server for testing recursive resolvers from the server side. It allows operators to craft custom DNS responses but is limited to a fixed set of templates and one-shot modifiers. These constraints prevent it from supporting large-scale, stateful, or compositional measurements where responses depend on runtime context or multi-stage interactions. By comparison, PRN provides full flexibility and fine-grained control over response generation, which is essential for the measurements we conduct.

8 Conclusion

We present a systematic study of various practical aspects of DNS self-amplification attacks and develop the first measurement framework specialized for evaluating the risk of such attacks on the ODNS infrastructure. Our methodology enables accurate estimation of MAF for a small subset of recursors that represent structural single points of failure due to their extensive downstream dependencies. These results facilitate effective triage on which entities to focus on and in what order. We anticipate that DaLens and its future extensions will play an essential role in our journey toward achieving a highly reliable and available DNS infrastructure.

Ethical Considerations

Our research follows the stakeholder-based ethics analysis as recommended by the USENIX Security Submission Policies.

Stakeholders and Potential Impacts. The primary stakeholders in this work include:

- End users: Our measurements involve public DNS resolvers, which serve millions of Internet users. We carefully designed our methodology to ensure negligible impact on users' experience: each resolver receives only a very small number of requests, far below their regular workload, and no sensitive user data is collected.
- Operators and vendors: DNS operators and software vendors may be affected by the discovery of security risks in Self-Amplification Attacks (SAAs). We anticipate that disclosure of our findings strengthens operational awareness and mitigations.
- Research community: This work provides a new methodology for risk assessment without injecting disruptive attack traffic, advancing ethical research practices in Internet measurement.
- Society: By identifying weaknesses in critical Internet infrastructure and proposing ways to measure them safely, our work contributes to broader Internet security and resilience.

Ethical Principles. We considered the Menlo Report principles:

- Beneficence: The benefits of proactively identifying systemic risks to resolvers outweigh the negligible measurement costs.
- Respect for Persons: We provide transparency via an informative website (https://dalens.netsec.ethz.ch/) under domains used in our study, and operators may opt out at any time.
- Justice: Our measurement does not target specific regions or operators disproportionately, and we collaborate with a national security authority to disseminate advisories globally.
- Respect for Law and Public Interest: All experiments are conducted using domains under our full administrative control, ensuring compliance with DNS standards and laws.

Harms and Mitigations. Potential harms include (i) minor additional load on resolvers and (ii) the risk of operators misinterpreting benign requests as anomalous. To mitigate these, we:

- Limited the request rate to negligible levels compared to normal workloads.
- Used only controlled domains to avoid unintended third-party data collection.
- Set up a publicly accessible website and opt-out mechanism for transparency and operator control.
- Coordinated with a national security authority to responsibly disseminate results and advisories.

Decision to Proceed and Publish. Balancing potential harms and benefits, we determined that the net ethical outcome is strongly positive: our approach enables proactive risk assessment without generating attack traffic, while providing actionable intelligence to operators and vendors. Both consequentialist reasoning (benefits to security outweigh minimal risks) and rights-based reasoning (respect for operators' autonomy via opt-out and transparency) support this decision. We consider both the research and its publication ethically justified.

Open Science

In accordance with the USENIX Security 2026 Open Science policy, we provided all artifacts necessary to evaluate our work. A permanent reference to the artifacts repository is provided at https://doi.org/10.5281/zenodo.17832418. These artifacts include the source code and scripts used in our experiments. Our release plan is carefully designed to balance enabling broad community benefit and the responsibility to prevent abuse.

Specifically, the tooling will not be made publicly available immediately. Access will be restricted to verified operators, vendors, and researchers and granted only upon request for legitimate purposes, such as evaluating their own software or deployments or conducting further academic research. Any potential broader release will be thoroughly reviewed in consultation with the institutional IRB to ensure that all ethical and security considerations are properly addressed.

At the same time, to enable the wider community to benefit from our work, we plan to evolve our current information page into a continuous, ethics- and privacy-aware SaaS-based platform dedicated to monitoring the prevalence and evolution of vulnerable behaviors across the global resolver ecosystem over time. It will publish only aggregated and anonymized statistics, allowing the research and operational communities to observe long-term trends without exposing sensitive identifiers. Verified entities that can demonstrate ownership of specific resolvers will also be able to submit targeted self-assessment tasks to validate or monitor their configurations. Through this approach, we ensure that the broadest community can benefit from our research while minimizing the risk of misuse or facilitation by attackers, ultimately fostering sustained collaboration and strengthening the resilience of the DNS ecosystem.

Further details will be provided on our information page (https://dalens.netsec.ethz.ch/).

References

[1] IPinfo Lite. https://ipinfo.io/lite. Accessed: 2025-08-11.

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

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

[4] Akamai. Whitepaper: DNS Reflection, Amplification, & DNS Water-torture. Technical report, 2019.

[5] Marios Anagnostopoulos, Georgios Kambourakis, Panagiotis Kopanos, Georgios Louloudakis, and Stefanos Gritzalis. DNS amplification attack revisited. *Computers & Security*, 39:475–485, 2013.

[6] Protick Bhowmick, Md Ishtiaq Ashiq, Casey Deccio, and Taejoong Chung. TTL Violation of DNS Resolvers in the Wild. In *International Conference on Passive and Active Network Measurement*, 2023.

[7] Stéphane Bortzmeyer, Ralph Dolmans, and Paul E. Hoffman. DNS Query Name Minimisation to Improve Privacy. RFC 9156, November 2021.

[8] Jonas Bushart and Christian Rossow. DNS Unchained: Amplified Application-Layer DoS Attacks Against DNS Authoritatives. In *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*, 2018.

[9] 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 Choffnes and Marinho Barcellos, editors, *Passive and Active Measurement*, 2019.

[10] Huayi Duan, Marco Bearzi, Jodok Vieli, David Basin, Adrian Perrig, Si Liu, and Bernhard Tellenbach. CAMP: Compositional Amplification Attacks against DNS. In *Proceedings of the USENIX Security Symposium*, 2024.

[11] Elias Heftrig, Haya Schulmann, Niklas Vogel, and Michael Waidner. The KeyTrap Denial-of-Service Algorithmic Complexity Attacks on DNS. Technical report, 2024.

[12] P. Hoffman and P. McManus. DNS Queries over HTTPS (DoH). RFC 8484, IETF, October 2018.

[13] P. Hoffman, A. Sullivan, and K. Fujiwara. DNS Terminology. RFC 8499, IETF, January 2019.

[14] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. Specification for DNS over Transport Layer Security (TLS). RFC 7858, IETF, May 2016.

[15] C. Huitema, S. Dickinson, and A. Mankin. DNS over Dedicated QUIC Connections. RFC 9250, IETF, May 2022.

[16] Geoff Huston. The Root of the DNS. *Internet Protocol Journal*, 28(2):14–30, 2025.

[17] 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.

[18] 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.

[19] Xiang Li. XMap: The Internet Scanner. https://github.com/idealeer/xmap. Accessed: 2025-08-11.

[20] Baojun Liu, Chaoyi Lu, Haixin Duan, Ying Liu, Zhou Li, Shuang Hao, and Min Yang. Who Is Answering My Queries: Understanding and Characterizing Interception of the DNS Resolution Path. In *Proceedings of the USENIX Security Symposium*, 2018.

[21] 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 the ACM SIGCOMM Conference*, 2023.

[22] Meng Luo, Liling Xin, Yepeng Yao, Zhengwei Jiang, Qiuyun Wang, and Wenchang Shi. Who Are Querying For Me? Egress Measurement For Open DNS Resolvers. In *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 1544–1550, 2023.

[23] Jonathan Magnusson, Moritz Müller, Anna Brunstrom, and Tobias Pulls. A second look at DNS qname minimization. In *International Conference on Passive and Active Network Measurement*, 2023.

[24] Florian Maury. The "indefinitely" delegating name servers (iDNS) attack. https://indico.dns-oarc.net/event/21/contributions/301/attachments/272/492/slides.pdf, 2015. Accessed 2022-04-30.

[25] 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.

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

[27] Giovane CM Moura, John Heidemann, Ricardo de O Schmidt, and Wes Hardaker. Cache Me If You Can: Effects Of DNS Time-To-Live. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2019.

[28] Giovane CM Moura, Ricardo de O Schmidt, John Heidemann, Wouter B de Vries, Moritz Muller, Lan Wei, and Cristian Hesselman. Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2016.

[29] Marcin Nawrocki, Maynard Koch, Thomas C Schmidt, and Matthias Wählisch. Transparent forwarders: an unnoticed component of the open DNS infrastructure. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, pages 454–462, 2021.

[30] ORYXLABS. PolarDNS. https://github.com/oryxlabs/PolarDNS. Accessed: 2025-12-09.

[31] Jeman Park, Rhongho Jang, Manar Mohaisen, and David Mohaisen. A large-scale behavioral analysis of the open DNS resolvers on the internet. *IEEE/ACM Transactions on Networking*, 30(1):76–89, 2021.

[32] Jeman Park, Aminollah Khormali, Manar Mohaisen, and Aziz Mohaisen. Where Are You Taking Me? Behavioral Analysis of Open DNS Resolvers. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019.

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

[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] Quirin Scheitle. MassDNS: A high-performance DNS stub resolver. https://github.com/blechschmidt/massdns. Accessed 2024-01-10.

[36] Kyle Schomp, Tom Callahan, Michael Rabinovich, and Mark Allman. On Measuring The Client-Side DNS Infrastructure. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2013.

[37] Raffaele Sommese, KC Claffy, 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.

[38] Petr Špaček. DNS Benchmarking 101: Essentials and Common Pitfalls. https://indico.dns-oarc.net/event/48/contributions/1033/attachments/991/1943/pspacek.pdf, Feburary 2024.

[39] Wenhao Wu, Zhaohua Wang, Qinxin Li, Zihan Li, Yi Li, Jin Yan, and Zhenyu Li. ODNS Clustering: Unveiling Client-Side Dependency in Open DNS Infrastructure. In *Proceedings of the ACM on Web Conference 2025*, pages 4745–4754, 2025.

[40] Chengxi Xu, Yunyi Zhang, Fan Shi, Hong Shan, Bingyang Guo, Yuwei Li, and Pengfei Xue. Measuring the Centrality of DNS Infrastructure in the Wild. *Applied Sciences*, 13(9):5739, 2023.

[41] 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.

[42] Ramin Yazdani, Mattijs Jonker, and Anna Sperotto. Swamp of Reflectors: Investigating the Ecosystem of Open DNS Resolvers. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)*, 2024.

[43] Ramin Yazdani, Yevheniya Nosyk, Ralph Holz, Maciej Korczyński, Mattijs Jonker, and Anna Sperotto. Hazardous Echoes: The DNS Resolvers that Should Be Put on Mute. In *2023 7th Network Traffic Measurement and Analysis Conference (TMA)*, 2023.

[44] 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 Measurement (PAM)*, 2022.

A Example Zone Configurations for Primitives

```
> zone a.com @ 1.2.3.4
q.a.com.     NS     f1.a.com
q.a.com.     NS     f2.a.com
q.a.com.     NS     f3.a.com
q.a.com.     NS     f4.a.com
...
```

(a) Fanout

```
> zone a.com @ 1.2.3.4
q.a.com.     NS     r1.a.com
r1.a.com.    NS     r2.a.com
r2.a.com.    NS     r3.a.com
r3.a.com.    NS     r4.a.com
...
```

(b) Referral

```
> zone a.com @ 1.2.3.4
q.a.com.     CNAME    c1.a.com
c1.a.com.    CNAME    c2.a.com
c2.a.com.    CNAME    c3.a.com
c3.a.com.    CNAME    c4.a.com
...
```

(c) Rewrite

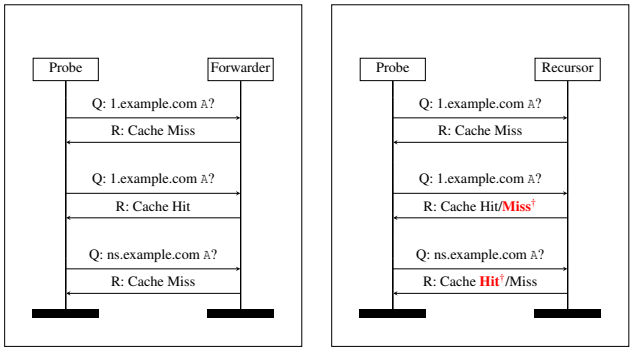
```
> zone a.com @ 1.2.3.4
d1 NS ns.d1.a.com
> zone d1.a.com @ 2.3.4.5
d2 NS ns.d2.d1.a.com
> zone d2.d1.a.com @ 3.4.5.6
d3 NS ns.d3.d2.d1.a.com
```

(d) DDLG

Figure 16: Zonefile examples for amplification primitives.

B Dependency Ordering

While egress anchors provide a stable mapping between vantage point–server pairs and egress sets, they do not yet reveal the structural dependencies between recursors and their downstream forwarders. In ODNS infrastructures, a server observed at the ingress may either be a recursor or a forwarder that relies on one or more upstream recursors. To attribute egresses to the correct recursors, we must infer a *dependency ordering* that captures the forwarder-to-recursor relationships.



(a) Forwarder (b) Recursor

Figure 17: Cache behavior observed for forwarders and recursors.

Directly distinguishing recursors from forwarders based solely on end-to-end observations is challenging. Theoretically, recursors perform full recursive resolution, whereas forwarders merely relay queries; however, when viewed only through latency and answer content, the differences can be subtle and easily masked by network delay. Our methodology therefore exploits the key behavioral divergence in *caching*.

Assertion 1: Cache Exclusivity

Only recursors retain intermediate resolution results; forwarders observe only final answers.

A recursor may cache both the final answer and any intermediate results encountered during recursive resolution (including NS and A/AAAA records). By contrast, a forwarder can only return cached final answers. Thus, if a request for a subdomain triggers a cache hit on an intermediate record, the server must be a recursor. In practice, we classify a server as a recursor if either (i) the second step of our probing sequence yields a cache miss, or (ii) the third step produces a cache hit. The comparison between the forwarders and recursors is shown in Figure 17. This criterion is robust to partial caching behavior, including the presence of frontend or backend caches on recursors as discussed in prior work [33].

Assertion 2: Latency Asymmetry

Recursive resolution exhibits higher latency than cache-based responses, while the round-trip times remain relatively stable across queries.

Latency provides a complementary signal for dependency inference. When a recursor must perform full recursive resolution, the request exhibits measurably higher latency. If a forwarder is queried directly, its latency profile reflects the upstream recursor's behavior, and does not itself show cache-driven acceleration. Combining cache observability and latency asymmetry enables us to infer the forwarder–recursor dependency: forwarders are those that never exhibit intermediate cache hits and whose latency patterns match pass-through behavior, while recursors are identified by their cache-exclusive responses and latency drops in repeated probing.

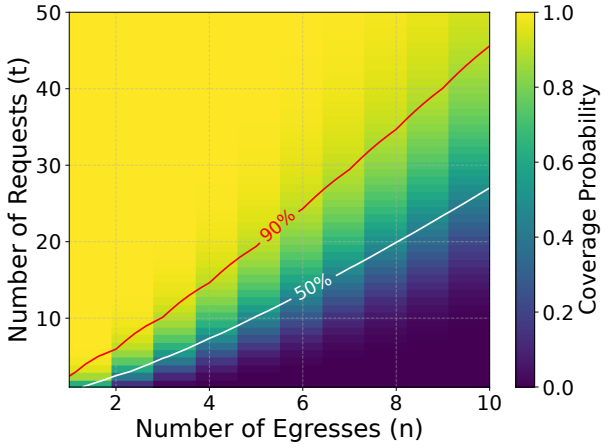


Figure 18: Egress Coverage Probability (Poisson Approx.).

C Egress Coverage Analysis

In practical DNS deployments, a single recursor may expose multiple egress IPs due to load-balancing mechanisms. Each

measurement request sent to the recursor is *randomly dispatched* to one of its n egress IPs with uniform probability.

Consider a recursor with n distinct egresses and t measurement requests. Let $P_{\text{cov}}(n,t)$ denote the probability that *all n egresses are observed at least once* among the t requests. This is equivalent to the classical *coupon collector problem* with replacement, where each draw corresponds to a request and each egress is a coupon type. By the inclusion–exclusion principle, we have:

$$P_{\text{cov}}(n,t) = \frac{1}{n^t} \sum_{k=0}^{n} (-1)^k \binom{n}{k} (n-k)^t. \tag{1}$$

Here, $\binom{n}{k}$ counts the subsets of size k egresses that are never selected, $(n-k)^t$ counts the sequences of requests using only the remaining $n-k$ egresses, and n^t is the total number of request sequences.

A Poisson-based approximation is shown in Figure 18, illustrating how the coverage probability increases with t for different numbers of egresses n. For small n, the probability rises rapidly with t, whereas for larger n a substantially greater number of requests is required to achieve the same coverage. For example, for a recursor with $n = 5$ egresses, approximately $t = 19$ distinct requests are needed to reach 90% coverage, while $n = 10$ would require close to $t = 45$ requests.

D Quantitative Analysis of Measurement Query Load

We follow the same terminology (i.e., request and query) as in the paper and assume that resolvers have already cached the authoritative records for the root and TLD zones. We then compute the total query load per stage below.

Discovery. We conducted three rounds of discovery measurements, each using four probes. Each probe sent four requests (two UDP and two TCP) of the form a.dalens.fun. Consequently, each resolver received a total of 3×4×4=48 requests and issued 3×4×4×2=96 queries in total across the three rounds.

Recursor Identification. Following Appendix C, we performed 45 measurement rounds (t), each using four probes. Every probe sent one request (preferably UDP if supported) with a QNAME of the form 10.9.8.7.6.5.4.3.2.1.conf-x-insy.dalens.fun, designed to trigger diverse egresses. The request rate was strictly rate-limited to below 5 QPS. Even if all queries induced by these 5 requests were issued within one second, the query rate would remain below 60 QPS, and in practice, they are further spread over time due to resolution latency. Each resolver thus received 45×4=180 requests in total and issued 45×4×12=2160 queries in 36 seconds.

Recursor Configuration Probing. We measured four attack primitives, using a single probe per recursor (from the anchor vantage point). Since a binary search was used to determine thresholds, the search space was 1–50 for NS fetch, NS chain, and CNAME chain (worst case 6 steps), and 1–100 for DDLG depth (worst case 7). Thus, each resolver received 6+6+6+7=25 requests in total in the worst case. The corresponding number of induced queries depended on the search steps and target values, with observed maxima of 253 and 603 and expected averages of 123.28 and 292.15 across all requests over time, respectively. The request rate was capped at 1 QPS per resolver; thus, the query rate would remain below 50/100 QPS. Furthermore, we followed a breadth-first scheduling–probing different resolvers before iterating on the same one, to further spread the load.

Amplification Power Estimation. Each resolver received exactly one request from one probe. The resulting number of induced queries varied depending on the resolver's vulnerability. Our measurements showed that 90% of resolvers issued fewer than 100 queries, about 2% exceeded 200, and fewer than 120 resolvers exceeded 300. As each resolver was probed only once rather than continuously, and even the largest induced query volumes are negligible compared to the steady-state traffic routinely handled by production resolvers, the measurement imposes no operational impact.

Overall Assessment. Across all measurements, each resolver saw only a minimal volume of probe traffic, spread over multiple rounds under strict rate limiting. For context, even a small deployment, such as a 2-core 4 GB virtual machine running on an Intel Xeon Platinum 8358 CPU, can handle well above 20,000 cache-miss queries per second. In practice, open resolvers that serve large user populations are provisioned with far greater capacity and redundancy. Therefore, the aggregate query load introduced by our measurements is negligible in both rate and volume, posing no practical impact on the tested resolvers or their upstream infrastructure.

E Resolution Configuration

E.1 Concurrent Forwarding

In this section, we show how DaLens leverages PRN's flexibility to expose *concurrent forwarding*. When a probe issues a single request, multiple upstream resolvers may attempt to resolve it simultaneously, each sending the identical QNAME. If all such queries were bound to one configuration, the PRN would map them to the same counter, producing state inconsistencies and incorrect results.

We design an *entry-point resolution configuration* that contains multiple CNAME templates in `conf-1`, each redirecting to a different configuration ID.

This example (Figure 19) illustrates how PRN's programmability supports fine-grained measurement tasks that are infeasible with conventional authoritative servers. The ability to define the entry point for resolutions with multiple

Figure 19: Resolution Configuration for Detecting Concurrent Forwarding.

dynamic branches provides researchers with a powerful tool to capture resolver behaviors that would otherwise remain hidden.

E.2 MAF Measurement

The resolution configurations used for the MAF measurements are shown in Figure 20. For example, to measure fanout-based message amplification, we configure the PRN with multiple glueless NS records, each redirecting to a different configuration. At the PRN, we log the source IP, instance ID, and configuration ID of every incoming query as shown in Figure 3. By correlating queries with the same instance ID, we reconstruct the full set of lookups triggered by a single query session. We then count how many distinct NS resolutions were observed for that instance, which reveals the fanout amplification factor.

Figure 20: Resolution Configuration for Primitives.