TsuNAME vulnerability and DDoS against DNS

ISI Technical Report ISI-TR-740
13 May 2021
Update from 6 May 2021
https://tsuname.io

Giovane C. M. Moura (1)   Sebastian Castro (2)   John Heidemann (3)
Wes Hardaker (3)
1: SIDN Labs   2: InternetNZ   3: USC/ISI

ABSTRACT

The [1] Internet's Domain Name System (DNS) is one of the core services on the Internet. Every web page visit requires a series of DNS queries, and large DNS failures may have cascading consequences, leading to unreachability of major websites and services. In this paper we present TsuNAME, a vulnerability in some DNS resolvers that can be exploited to carry out denial-of-service attacks against authoritative servers. TsuNAME occurs when domain names are misconfigured with cyclic dependent DNS records, and when vulnerable resolvers access these misconfigurations, they begin looping and send DNS queries rapidly to authoritative servers and other resolvers (we observe up to 5.6k queries/s). Using production data from `.nz`, the country-code top-level domain (ccTLD) of New Zealand, we show how only two misconfigured domains led to a 50% increase on overall traffic volume for the `.nz`'s authoritative servers. To understand this event, we reproduce TsuNAME using our own configuration, demonstrating that it could be used to overwhelm *any* DNS Zone. A solution to TsuNAME requires changes to some recursive resolver software, by including loop detection codes and caching cyclic dependent records. To reduce the impact of TsuNAME in the wild, we have developed and released `CycleHunter`, an open-source tool that allows for authoritative DNS server operators to detect cyclic dependencies and prevent attacks. We use `CycleHunter` to evaluate roughly 184 million domain names in 7 large, top-level domains (TLDs), finding 44 cyclic dependent NS records used by 1.4k domain names. However, a well motivated adversary could easily weaponize this vulnerability. We have notified resolver developers and many TLD operators of this vulnerability. Working together with Google, we helped them in mitigate their vulnerability to TsuNAME.

1 INTRODUCTION

The Internet's Domain Name System (DNS) [18] provides one of the core services of the Internet, by mapping hosts names, applications, and services to IP addresses and other information. Every web page visit requires a series of DNS queries, and large failures of the DNS have severe consequences that make even large websites and other internet infrastructure fail. For example, the Oct. 2016 denial-of-service (DoS) attack against Dyn [4] made many prominent websites such as Twitter, Spotify, and Netflix unreachable to many of their customers [29]. Similarly, a DDoS against Amazon's DNS service affected service for a large number of services [45] in Oct. 2019.

The DNS can be seen as a hierarchical and distributed database, where DNS *records* [19] are stored in and distributed from *authoritative servers* [10] (for instance, the Root DNS servers [38] distribute records from the Root DNS zone [39]). As such, all information about an end domain name in the DNS are served by *authoritative servers* for that domain. This information is typically retrieved by *recursive resolvers* [10], which answer questions originally posed by users and their applications. Recursive resolvers are typically operated by a user's ISP, or alternatively public DNS resolvers operated by Google [8], Cloudflare [1], Quad9 [31], Cisco OpenDNS [27] and others.

The configuration of authoritative servers and their records is prone to several types of errors [2, 16, 18, 28, 42]. In specific, *loops* can be introduced while setting authoritative DNS servers for delegations– either using CNAME records (§3.6.2 in [18]) or NS records (§in 2 [16]), also known as cyclic dependencies [28]). While such loops existence has been documented, in this work we show how they can be *weaponized* to cause DDoS.

In specific, we examine the case of *cyclic dependency*, an error which occurs when NS records for two delegations point to each other. Since NS records define authoritative servers used to resolve a domain [18], when a cyclic dependency occurs, neither name can be definitively resolved. For example, suppose that the NS record of `example.org` is `cat.example.com`, and the NS record of `example.com` is `mouse.example.org`. This misconfiguration (`example.org`↔`example.com`) creates a situation in which resolvers cannot retrieve the IP

[1] This version is an update from the original version of May 6th, 2021. It clarifies the relationship between TsuNAME and RFC1536, and adds §6, which covers the threat model.

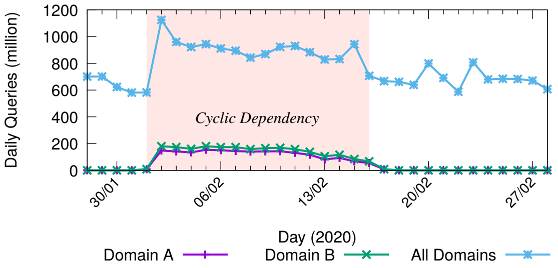


Figure 1: Queries timeseries for all domains and cyclic dependent domains (A and B)

addresses associated with NS records for either zone. Without these addresses, recursive resolvers are unable to answer any questions from their clients about that portion of the DNS tree below these delegation points[2].

The first contribution of this paper is to report that, in the wild, cyclic dependencies can result in a query cascade that greatly increases traffic to authoritative servers. An example of this problem is the `.nz` event (§2). On 2020-02-01, a *configuration error* (i.e., not an intentional attack) resulted in two domains being misconfigured with cyclic dependent NS records. That, in turn, was followed by a 50% traffic volume surge to the the authoritative servers for the country-code top-level domain (ccTLD) of New Zealand (`.nz`), from 800M to 1.2B daily queries (shaded area in Figure 1). This event did not disrupt the `.nz` authoritative servers. However, the same misconfiguration can also lead to even far more queries, depending on the domain and TLD: after we disclosed this vulnerability to TLD operators , an European ccTLD shared with us that it had experience 10x traffic growth after two domains where misconfigured with cyclic dependencies .

These examples bring us to question: *what would happen if an attacker would intentionally misconfigure hundreds of domains this way, at the same time*? The `.nz` event demonstrates what only two domains can do, but a motivated attacker could cause a far larger traffic surge, which could ultimately overwhelm authoritative servers, affecting *all* their users, and possibly affecting additional zones due to collateral damage . This poses a great concern for any domains and registrations points, such as TLDs and ccTLDs. Critical domains, and ccTLDs in particular, frequently provide essential services to their users, such as access to government services, banking and on-line shopping.

Our second contribution is to demonstrate this threat in controlled conditions in §3. We *emulate* a TsuNAME event by setting up multiple cyclic dependent domain names under our control on our servers (so as to not harm others) and measure the consequences. We show that Google's public DNS (GDNS) is responsible for the bulk of queries, but we also found other vulnerable resolvers in 260 Autonomous Systems (ASes). Following responsible disclosure practices, we notified Google and other TLD and resolver operators . Google and OpenDNS, have already fixed their software.

Our final contribution is to develop `CycleHunter`, a tool that finds cyclic dependencies in DNS zone files (§4). This tool allows *authoritative* server operators (such as ccTLD operators) to identify and mitigate cyclic dependencies, *preemptively* protecting their authoritative servers from possible TsuNAME attacks. We use `CycleHunter` to evaluate the Root DNS zone and 7 other TLDs (~185M domain names altogether), and found cyclic dependent domains in half of these zones.

We made `CycleHunter` publicly available at Github and we thank the various contributors that have helped improve the tool. We have carefully disclosed our findings with the relevant DNS communities .

2 .NZ EVENT

On 2020-02-01, two domains (`DomainA` and `DomainB`) under `.nz` had their NS records misconfigured to be cyclically dependent. `DomainA` NS records were pointed to `ns[1,2].DomainB.nz`, while `DomainB` NS records pointed to `ns[1,2].DomainA.nz`. This configuration error led to a 50% surge in query volume at `.nz` authoritative servers (Figure 1)[3]. The `.nz` operators manually fixed this misconfiguration on 2020-02-17, after which the queries to return to normal levels.

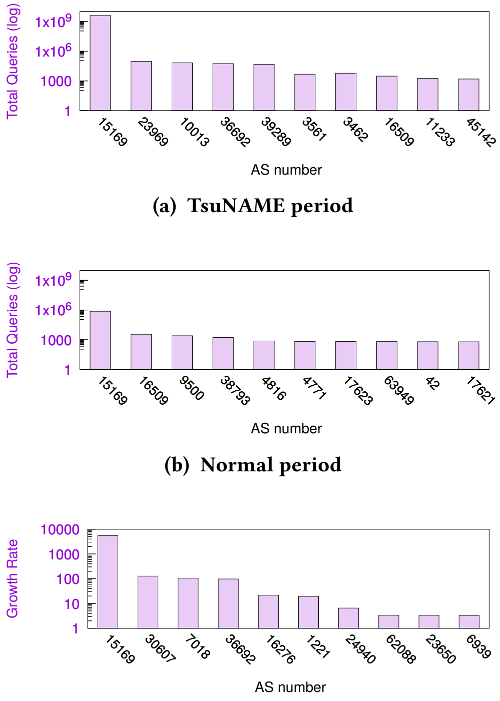
2.1 Query sources

During the sixteen day period of the TsuNAME event (2020-02-[01–17]), there were 4.07B combined queries for `DomainA` and `DomainB`, with a daily average of 269M. Figure 2a shows the top 10 ASes by query volume during the event period. The overwhelming majority (99.99%) of the traffic originated from Google (AS15169), with only 324k queries from 579 other ASes – the queries from Google outnumbered the other ASes by 4 orders of magnitude.

For comparison, Figure 2b shows the top 10 Ases for both domains during the "normal" periods when there was no cyclic dependency, spanning over the 16 days before and after the TsuNAME period (2020-01-[24–30] and 2020-02[18–28]). During this "normal" period, Google sent no more than 100k daily queries for both `DomainA` and `DomainB`. During the TsuNAME period, however, Google's query volume multiplied 5453× (Figure 2c). No other AS had an traffic growth larger than 100x in the same period. (Google operates Google

[2] Although parent authoritative servers provide IP addresses for NS records within a child domain (known as glue records), they cannot provide them for NS records that exist in other zones.

[3] Our vantage point – the authoritative DNS servers of `.nz`– sees only queries from DNS resolvers and not directly from end users or forwaders), given that most users do not run their own resolvers and instead use their ISP's or pubic DNS resolvers, such as the Quads1,8,9.



(a) TsuNAME period

(b) Normal period

(c) Growth rate

Figure 2: Top 10 ASes querying for Domains A and B, for .nz. TsuNAME period: Feb. 1–17, normal period: Jan. 24–30, Feb. 18–28, 2020.

Public DNS – GDNS – a large, popular public resolver service [8] and makes up 8% of *all* queries sent to .nz [21]).

2.2 Scrutinizing Google queries

The question *Why was Google solely responsible for most of the queries?* relates to two others: *How long should resolvers retry when resolving domains with cyclic dependencies?* And *how aggressive should be they be when finding answers?*

Previous research has shown resolvers will *hammer* unresponsive authoritative servers [25] – with up to 8 times more queries, depending on the DNS records' time-to-live (TTL) value. But in the case of cyclic dependencies, authoritative servers are *responsive* and resolvers *bounce* from one authoritative server to another, asking the same sequence of questions repeatedly. As such, cyclic dependency is different from the (partial) *unresponsiveness* situation from [25].

Given that Google was responsible for virtually all queries during the .nz TsuNAME event for the cyclic dependent domains (§2.1), we isolate and study the queries from Google. Table 1 shows the breakdown of the query names and types from Google during the .nz event. We see that most queries to .nz are for A and AAAA records for the two domain's own NS records (NS records store the authoritative server names of a domain, while A [18] and AAAA records [44] store each server's IPv4 and IPv6 addresses, respectively. These queries, however, can never be resolved in this cyclic dependent scenario, as one authoritative server keeps forwarding resolvers to the other. The NS records, however, were readily available within the .nz zone – which explains the lower volume of queries.

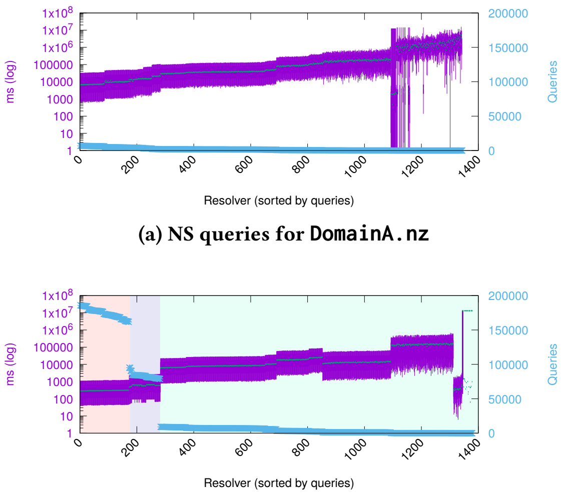
Query Name	Query Type	Queries(v4)	Queries(v6)
DomainA.nz	NS	13.0M	10.9M
DomainB.nz	NS	4.3M	3.0M
ns1.DomainA.nz	A	266.1M	281.3M
	AAAA	266.2M	281.4M
ns2.DomainA.nz	A	266.1M	281.2M
	AAAA	266.1M	281.4M
ns1.DomainB.nz	A	222.6M	237.9M
	AAAA	222.5M	237.7M
ns2.DomainB.nz	A	222.5M	237.7M
	AAAA	222.3M	237.5M

Table 1: Google queries during the .nz event

2.2.1 Interquery interval. How frequently did GDNS resolvers send .nz queries for these domains during the TsuNAME event? To measure this, we pick a date (2020-02-06) during the period and compute the inter-query interval time, *i.e.,* the time in-between two queries arriving from the same IP address to the .nz authoritative servers for the same query name and type.

Figure 3 shows the results (for space constraints, we show only results for the queries highlighted in the green rows of Table 1). We start with the NS queries to DomainA.nz. Figure 3a shows individual resolvers on the x axis, and number of queries on they sent on the right y axis. We see that all resolvers send fewer than 10k queries. On the left y axis, we show the interval inter-quartile range (IQR) of the time between queries (with the green line showing the median value in ms). Given that the TTL value of these records is 86400 s (1 day), we should not see any resolver sending more than one query on this date (anycast-based resolvers cache has multiple layer of complexity, and are not always shared [25, 32]).

As shown in Table 1, the bulk of queries is for A and AAAA records of the authoritative servers of DomainA and DomainB . Figure 3b shows the results for A records of ns1.DomainA.nz. We see three categories of resolvers, according to their query volume, which we highlight in different colors. The first group – *heavy hammers* – sent 162-186k queries on this day, one every 300 ms. The second group – *moderate hammers* – sent 75-95k daily queries, one every 590 ms – roughly the double of the heavy hammers. The last group, which covers most of the addresses – is less aggressive: they sent up to 10k daily queries each. Given they are more numerous than the other group, their aggregated contribution matters. (**??**



(a) NS queries for DomainA.nz

(b) A queries for ns1.DomainA.nz

Figure 3: Google (AS15169) resolvers on 2020-02-06, during .nz TsuNAME event: time in between queries.

	Zones	
	sub.verfwinkel.net	sub.cachetest.net
NS	ns.sub.cachetest.net	ns.sub.verfwinkel.net
TTL	1s	1s

Table 2: New domain experiment setup.

shows the results for AAAA records, which are similar to Figure 3b).

This heterogeinity in Google's resolver behavior is surprising. We notified Google and were able to work with them on the issue, and they both confirmed and fixed their Public DNS service on 2020-02-05.

3 EMULATING TSUNAME

3.1 New domain experiment

In this first experiment, we are interested in determining the *lower* bound in queries that authoritative servers can experience during a TsuNAME event, by using domain names never used beforehand, so they would not have been cached or have a query history.

Setup: we configure two third-level domains with cyclic dependencies (Table 2). We use third-level instead of second-level domains given it is the authoritative servers of the parent zone that experience the traffic surge – if example.org is misconfigured, it is its parent .org authoritative servers that will see the traffic surge.

We ran our own authoritative servers using BIND9 [14], one of the most popular open-source authoritative server software, on Linux VMs in located in AWS EC2 (Frankfurt).

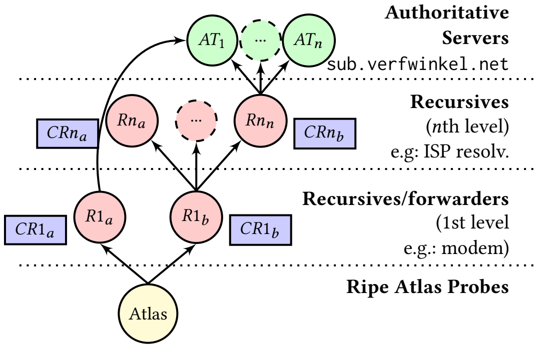


Figure 4: Relationship between Atlas probes(yellow), recursive resolvers (red) with their caches (blue), and authoritative servers (green).

To minimize caching effects, we set every DNS record with a TTL =1 s (Table 2). By doing that, we maximize the chances of cache miss, increasing total traffic we observe at our authoritative servers.

Vantage points (VPs): we use ∼10k Ripe Atlas probes [34, 35] as VPs. Ripe Atlas probes comprise a network of more than 11k active devices, distributed among 6740 ASes across the globe (Jan. 2021). They are also publicly accessible as well as the datasets of our experiments [33].

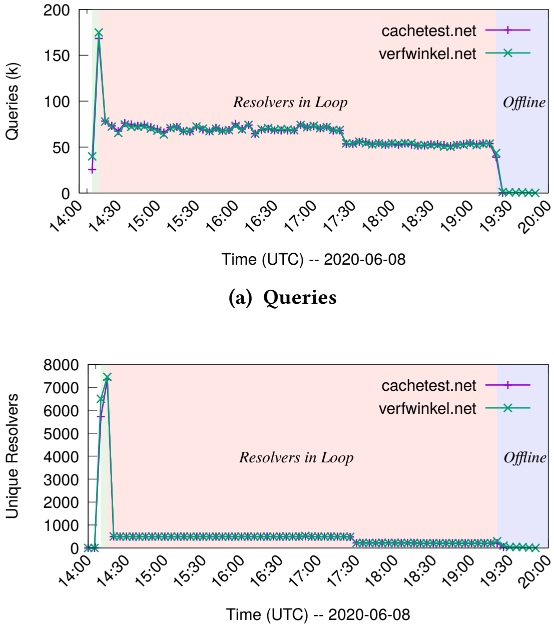
We configure each Atlas Probes to send *only one* A record query for PID.sub.verfwinkel.net., where PID is the probe unique ID [36]. By doing that, we reduced the risk of warming up the resolver's caches for other VPs. The query is sent to each probe's local resolver, as can be seen in Figure 4. As one probe may have multiple resolvers, we consider a VP as a unique probe ID and each of its resolvers.

3.1.1 Results: Table 3 shows the results for this measurement ("new domain" column). On the *client side*, *i.e.*, traffic measured between Atlas probes and their 1st level recursive resolvers (Figure 4), we see ∼9.7k Atlas probes that form 16.8k vantage points. Altogether, they send 18715 queries to their first level resolvers (retrieved from Ripe Atlas, publicly available at [33]), which are mostly answered as SERVFAIL [18] or they simply timeout – both status indicating issues in the domain name resolution.

Heavy traffic growth on the authoritative server side: on the authoritative server side (between authoritative servers and n_{th} level resolvers in Figure 4), we see ∼11k IPs addresses. As each Atlas probe query its resolver, their resolver, in turn, may forward the queries to other resolvers, and so forth [25, 32], – and our authoritative servers see only the *last* resolver in the chain. In total, these resolvers belong to ∼2.6k ASes, and ended up sending ∼ 8M queries to both authoritative servers - *435x more queries* than the client side.

Measurement	New Domain	
Frequency	once	
Qname	`$PID.sub.verfwinkel.net.`	
Query Type	A	
Date	2020-06-08	
Duration	6h	
Client Side		
Atlas Probes	9724	
VPs	16892	
Queries	18715	
Responses	18715	
SERVFAIL	12585	
Timeout	5969	
REFUSED	103	
FORMERR	28	
NOERROR	22	
NXDOMAIN	8	
NO ANSWER	0	
Authoritative Server Side		
	ns1	ns2
Querying IPs	11195	11572
ASes	2587	2611
Queries	4064870	4080446
Responses	4064801	4070035

Table 3: TsuNAME Emulation. Datasets: [33]



(a) Queries

(b) Resolvers

Figure 5: New domain measurement: queries and unique resolvers timeseries (5min bins)

Identifying problematic resolvers. Figure 5 shows the timeseries of both queries and resolvers we observe at our authoritative servers (each line shows a different authoritative server, one per domain). We identify three phases in this measurement: the first phase (green shaded area, x < 14:30 UTC, is the warmup phase: this is when the VPs send the queries we have configured. We see more than 150k (Figure 5a) arriving each authoritative server, from roughly 5k resolvers (Figure 5b).

After 14:30, however, Atlas probes stop sending queries to their 1st level resolvers. Even in the absence of Atlas probes, the authoritative servers keep on receiving queries from resolvers – as shown in the salmon area ("Resolvers in Loop"). We label these resolvers as *problematic*: they should not have being resending queries for hours and hours. In total, we see 574 resolvers from 37 Ases showing this looping behavior (New domain in Table 4).

The last phase (x >19:30) is when we stopped BIND9 on our authoritative servers, and kept on collecting incoming queries ("offline" phase). At that stage, our servers became *unresponsive*. Once the problematic resolvers cannot obtain any answers, they quickly give up and the number of queries reduce significantly. Without our manual intervention, one may wonder *when* these loops would stop (we show in §4.2 that these loops may last for *weeks*).

	Queries	Resolvers	ASes
New domain	7.5M	574	37
Recurrent	30.6M	1423	192
Sinkhole	18.1M	2652	127
Unique	56.2M	3696	261

Table 4: Problematic Resolvers found on experiments

Other ASes also affected: Figure 6 shows the histogram of queries per source ASes. We see than the #1 AS (Google, 15169) is responsible for most queries (~60%), a far more modest figure than on the .nz event (§2). We see that other ASes are also affected by the same problem: AS200050 (ITSvision) and AS30844 (Liquid Telecom) send both many queries too. In fact, we found in this experiment that 37 ASes were vulnerable to TsuNAME (Table 4).

How often do the problematic resolvers loop? For each problematic resolver, we compute the interval between queries for the query name and query type, for each authoritative server, as in §2.2. Figure 7 shows the top 50 resolvers that sent queries to one of the authoritative servers for A records of `ns.sub.cachetest.net`. We see a large variation in behavior. The first resolver ($x = 1$) sends a query every 13ms, and accumulated 858k queries during the "Resolvers in loop". The other resolvers (20< x <50) all belong to Google, and

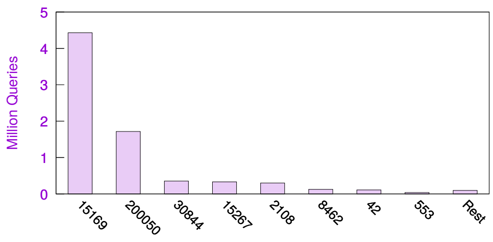


Figure 6: New domain: queries per AS with problematic resolvers

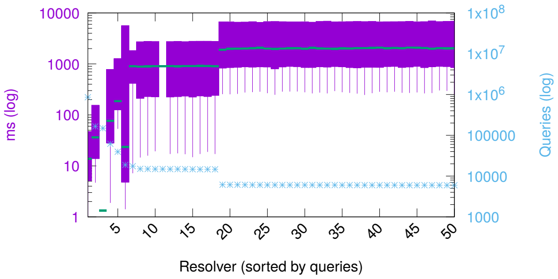


Figure 7: New domain: IQR and queries for A records of `ns.sub.cachetest.net`

have a more stable behavior; sending roughly the same number of queries over the same interval (median 2900 ms). As shown in Figure 6, taking altogether, Google resolvers are responsible for most of the queries, but they are not the most aggressive individually. Resolvers 7–19 loop every second, while resolvers 20–50 query on median every 3s – and the latter all are from Google.

4 DETECTING CYCLIC DEPENDENCIES

TsuNAME attacks are intrinsically asymmetrical: the victims (authoritative server operators) are typically different companies from the attackers (vulnerable resolvers operators). We discuss in more details the threat model in §6).

Next we assume the side of authoritative server operator, and work on *preventing* TsuNAME attacks, by detecting and removing cyclic dependencies from their zones. We present `CycleHunter`, a tool that we developed that *proactively* detects cyclic dependencies in zone files, and allow operators to discovery them *before* problematic resolvers do. We make `CycleHunter` publicly available at http://tsuname.io and [6].

`CycleHunter` uses active DNS measurements to detect cyclic dependencies, given many NS records in a DNS zone are typically out-of-zone (out-of-bailiwick) [42]. As such, it requires knowledge from external zones, which could only be done if an operator had in possession all necessary zone files (a condition we do not assume).

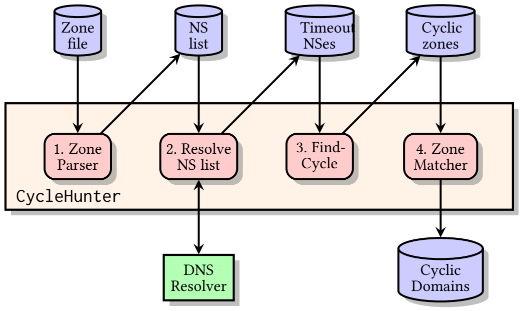


Figure 8: `CycleHunter` workflow

4.1 `CycleHunter`

Figure 8 shows `CycleHunter`'s workflow. It is divided in four main parts, which we describe next:

1. Zone Parser: we start with this module, that reads a DNS zone file, such as the the `.org` zone. Zone files contains delegations, and various types of records (A,AAA, NS, SOA, DNSSEC, and so forth). This module extracts *only* the NS records, outputting into a text file (NS List in Figure 8). Our goal is to determine *which* of these NS records are cyclic dependent, and, ultimately, what domains names in the zone file use those cyclic dependent NS records. Given many domain names use the same authoritative servers [3, 15], this step significantly reduces the search space. The `.com`, for instance has 151M domain names, but only 2.19M unique NS records (Table 5).

2. Resolve NS list: this modules tries to resolve every single NS record in the NS list. `CycleHunter` uses whatever resolver the computer is configured with (we use BIND 9 in our experiments), and queries for the start-of-authority (SOA) record [18], a record that every domain must have, for each NS in NS list. If a resolver is capable to retrieve a domain's SOA record, it means that the domain is *resolvable* and, as such, not cyclic dependent. Thus, we are interested only in domains that *fail* this test, given cyclic dependent NS records are not resolvable either. A NS record resolution from can fail for several reasons: the domain name does not exist (NXDOMAIN), lame delegations (the servers are not authoritative for the domain [17]), transient failures, and so forth.

3. Find Cycle: this module is the one that ultimately detects cyclic dependent NS records. This module tells cyclic depend zones from other types of errors. It starts by creating Authority objects (to store Authority Section DNS data [18]) for each NS in `NS list`. For example, suppose that `ns0.wikimedia.org` was in the NS list (Figure 9). This module then creates an `Authority` object for this NS record, which includes its parent zone `wikimedia.org` and its NS records (`wikimedia.org: [ns1,ns2].example,com`). It does that by querying the parent authoritative servers instead of the unresponsive cyclic NS

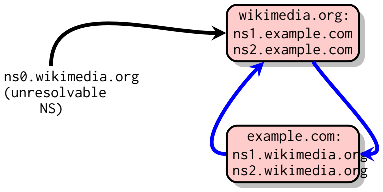


Figure 9: CycleHunter Cyclic Dependency Detector

records – in our example, it retrieves the authority data for wikimedia.org directly from the .org authoritative servers.

The next step consists in determining what *zones* this Authority zone depends, by analyzing its own NS records. In our fictional example, we see that wikimedia.org depends on example.com. So we also need to create an authority object for example.com, and determine what zones it depends on. The final step consist in comparing these two authority objects: as can be see in Figure 9, example.com NS records depend on wikimedia.org, which in turn depends on example.com, creating a cyclic dependency between wikimedia.org and example.com.

CycleHunter is also able to detect other types of dependencies. For example, if the zone wikimedia.org would have a in-zone NS record (ns3.wikimedia.org) but with a unresponsive or lame glue (or missing glue), CycleHunter would then classify this zone as cyclic dependent with in-zone NS record ("fullDepWithInZone").

Zone Matcher: the last module of CycleHunter tells which domains in the DNS zone use those cyclic dependent NS records found by Find Cycle. For example, ns0.wikipedia.org could have been the authoritative server for both dog.org and cat.org.

Performance: CycleHunter is a concurrent and asynchronous application that allows the user to set as a parameter the number of threads/workers. However, the bottleneck is actually the *resolver* used in Step 2. As such, we recommend operators to run a high performance resolver for faster results – and *always* clean the resolver's cache *before* running CycleHunter, to retrieve the most updated information.

4.2 DNS Zones Evaluation

We use CycleHunter to evaluate 7 TLDs and the Root DNS zone, which are either public [12, 13], or available via ICANN CZDS [11]. We show the zones in Table 5. For each zone, we show the number of domains (size) and the total number of NS records (NSset).

From the total 184M domains we evaluated, we obtained 3.6M distinct NS records. CycleHunter then probes each of them as described in Figure 8, and ultimately we found 44 cyclic dependent NS records (Cyclic in Table 5). We manually verified the 44 cyclic dependent records and confirmed the results. In total, 1435 domain names employed these cyclic dependent domains, and are ultimately unreachable.

zone	Size	NSSet	Cyclic	Affect.	Date
.com	151445463	2199652	21	1233	2020-12-05
.net	13444518	708837	6	17	2020-12-10
.org	10797217	540819	13	121	2020-12-10
.nl	6072961	79619	4	64	2020-12-03
.se	1655434	27540	0	0	2020-12-10
.nz	718254	35738	0	0	2021-01-11
.nu	274018	10519	0	0	2020-12-10
Root	1506	115	0	0	2020-12-04
Total	184409371	3602839	44	1435	

Table 5: CycleHunter: evaluated DNS Zones

The numbers, fortunately, are not that large, and suggest that they are more likely to be cause by configuration errors – as these domains are unresolvable. However adversaries could exploit that to incur damage.

4.2.1 Singling out .nl Cyclic Domains: The 6M .nl domain names yield to 79k NS records (Table 5). CycleHunter identified 6 zones related to these NSes that had cyclic dependency – 3 of them were .nl domain names, and the other were 2 .com and 1 .net. There were 64 domains that employed these cyclic DNS zones (affect.).

Out of the 3 .nl zones, two were test domains we configured ourselves – so they are not publicized and receive no more than 1k daily queries. The remaining domain (bugged-example.nl), however, is an old domain, registered in 2004.

Given we have access to .nl authoritative DNS traffic, we use ENTRADA [40, 46], an open-source DNS analytics platform to determine the daily queries this cyclic domain received. Figure 10 shows the results. We see very few queries until mid June (<300 daily). However, on May 19, the domain owner changed the NS records of the domain, to a cyclic dependent setup – probably a human error as in the case of .nz (§2). And from June 4th, we start to observe a significant amount of queries to this domain: 2.2M, reaching up to 27M on June 8th. From that point on, we see three intervals with large volume of queries, which average each 42M daily queries to this domain. The first interval (July 3rd–July 13th), last for the 10 days, the second for over amonth (Sep. 13th – Oct. 15th), an the last one for 43 days (Oct. 21st – Dec. 3rd).

Figure 10 shows also that most of these queries come from Google. To fix that, we notified the domain owner on Dec. 4th, and they quickly fixed their NS settings, which after that, the number of queries reduced to 300 daily (we did this analysis prior to GDNS being repaired .

We see that simply having cyclic dependencies does not trigger large volume of queries – our two test domains that have cyclic dependency have not experienced large volume of queries, but only in combination with vulnerable resolvers.

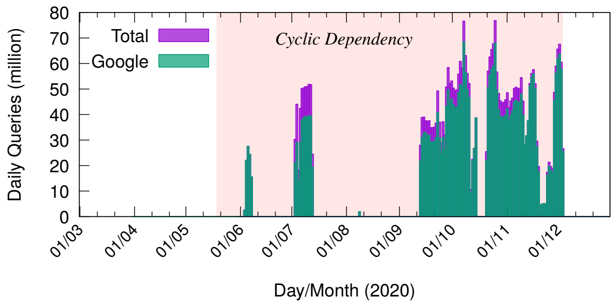


Figure 10: Query timeseries for `.nl` domain with cyclic dependency found with `CycleHunter`

4.3 Resolver software evaluation

Google reparing GDNS reduced the potential for attack harm. However, as shown in Table 4, Google was not the only affected resolver operator.

We set out to determine if current resolver software is vulnerable to TsuNAME. We set out two tests: determine if a resolver loops in the presence of cyclic dependencies and if it caches/detect these loops (https://tsuname.io/advisory.pdf.

We configure a test zone with cyclic dependency (as we did in §3) and evaluate popular DNS resolvers: Unbound (v 1.6.7) [26], BIND (v 9.11.3) [14], and KnotDNS (v 5.1.3) [7], on a VM we run on AWS EC2 (Fra). We found that none of the start looping in the presence of cyclic dependent domains, hence they are not vulnerable.

We also evaluated public DNS services, in specifically Quad9 [31], Quad1 [1]. However, we found that Cisco's OpenDNS [27] was also vulnerable. We notified the operators, and they have also fixed the issue on 2021-04-13.

5 RESPONSIBLE DISCLOSURE

We wish to protect zone operators from DDoS attacks with TsuNAME. The problem can be solved by modifying recursive resolvers and DNS forwarders to break from loop in the presence of cyclic dependencies, and by deploying updated versions of resolvers software (which always takes time). However, operator of an authoritative DNS service can protect their servers by ensuring that no zones in their domain have cyclic dependencies. In the long run, both of these changes should be in place, since either new recursive software or new domains with cyclic dependencies would otherwise recreate the problem.

To address these problems we have reached out to notify Google, whose public DNS service represents the largest recursive resolver traffic we see, and worked together with them. After that, they solved this problem at their GDNS. As part of these disclosures we followed best practices, allowing operators at least 90 days to address problems. This threshold is consistent with `cert.org`'s 45-day notification policy [5] Google Project Zero's 90-day policy [9].

Date	Type	Group
2021-02-05	Private Disclosure	OARC34
2021-02-22	Private Disclosure	APTLD
2021-02-23	Private Disclosure	CENTR
2021-03-04	Private Disclosure	LACTLD
2021-02-18–2021-05-05	Private Disclosure	Private
2021-05-06	Public Disclosure	OARC35
2021-05-06	Public Disclosure	https://tsuname.io

Table 6: TsuNAME disclosure timeline

In addition to Google we also notified operators of the ASes that generated the greatest amount of recursive traffic in our experiments from §3. Of these ten, three responded to us. Of those, two reported that they were running very old recursive resolver software. One was using PowerDNS resolver (3.6.2-2, from 2014 [30]), and the other was using Windows 2008R2. Both planed to update these resolvers.

5.1 Private and public disclosure

Our first private notification to a group was during OARC34, in which we disclosed the vulnerability during a members-only section(2021-02-05) [20], as can be seen in Table 6Moreover, we have disclosed the vulnerability to other trusted communities, including the Asian Pacific, the European, and the Latin American TLD associations (APTLD, CENTR, and LACTLD, respectively). We have also notified the Root DNS operators and Verisign, the operator of `.com` and `.net`. We will public disclose the vulnerability on May 6th, 2021 (3 months after the first private disclosure at DNS-OARC).

Operators reaction: Our presentation during the Virtual OARC34 meeting draw interest from various operators. First, we had publicly two other ccTLDs that had experienced this type of attack firsthand. The first one – an European ccTLD – went through the same as `.nz` and had its own event, which they kindly shared with us.

On a particular day in 2019, around 19:00 UTC, two domains in their zones were misconfigured with cyclic dependencies. Given these domain names were particularly popular in the country, it cause the largest surged we have seen from TsuNAME related events: 10x traffic growth. Figure 11 shows a timeseries of queries (y axis anonymized by the operator). It was only fixed once the ccTLD operator contacted the domain owner, who fixed the situation on the day after, around 11:00 UTC. Similarly to the `.nz` event, we see a immediate drop in the traffic.

A second large anycast operator confirmed that Google had at least in one occasion sent a surge of queries to their authoritative servers, several years ago, following a misconfiguration with cyclic dependency. Their experience was

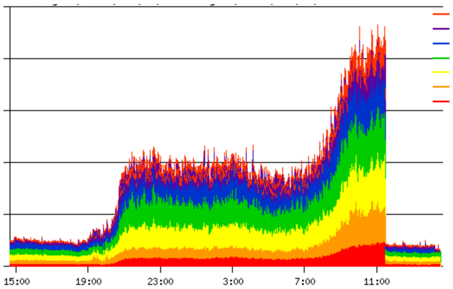


Figure 11: TsuNAME event at an Anonymous EU-based ccTLD operator.

worse than ours: similar to our experiments (§3), the first see a surge in UDP queries from Google. To curb that, they use response rate limiting (RRL), which allows an authoritative server to set the maximum queries/s, per client IP address [43]. When queries start not being answered, they observed that Google started to fall back over TCP, which has its own overhead compared to UDP, amplifying the problem even further. Last, several operators downloaded and contributed to `CycleHunter`, our open-source tool, making it faster and improving its usability. We are thankful to them.

Public Disclosure: On May 6th, we will public disclosed TsuNAME, on its official website [41] and also during DNS OARC 35. We relased both a security advisory and a technical report.

5.2 Problem solved at Google and OpenDNS

After our notifications, both Google and OpenDNS fixed their public resolver services. This, in turn, should reduce the volume of queries one may receive in an event that exploits TsuNAME (we confirmed this for Google Public DNS).

However, there are plenty of older resolver software that may be vulnerable on the Internet –see Figure 6 – and, as such we recommend both authoritative servers operators to take measuremes to prevent attacks. We describe what steps to take at a security advisory we release on May 6th, 2021 [23].

6 THREAT MODEL

The TsuNAME threat model involves in using *DNS reflection* to carry out a denial-of-service attack. Instead of attacking these servers directly, the attack could use cyclic dependent domains and vulnerable resolvers to keep a continuous stream of queries to the designated targets. None of our experiments fully exploited this possibility for ethical reasons; next we discuss how a well motivated could attack could as well do it.

For this to happen, an attacker needs (i) to have domains under a given zone (or take control over them, *e.g.,* by stealing registrant or registrar credentials), (ii) misconfigure them with cyclic dependent NS records, and (iii) induce vulnerable resolvers to carry out queries.

The first and second part are not difficult – most TLDs such as `.org` and `.es` have an open registration policy, so anyone can register domains, and misconfiguring them. For example, say an attacker has 500 `.fr` and 500 `.ca` domain names under its disposable: it could configure each of them with NS records pointing to each other, as Figure 9.

The last step consists in inducing vulnerable resolvers to query for these domains, so they can enter start looping and unleash a large volume of queries. It will be the parent authoritative servers of the cyclic NS records that will be receiving all the queries (in this case, `.ca` and `.fr` authoritative servers).

The last step involves in finding vulnerable resolvers – our experiments show that there are 4k resolves from 261 Ases vulnerable to TsuNAME, but that is a lower-bound estimative, given we have not covered most resolvers on the Internet (we were limited by the view of our vantage points). Luckily, Google has fixed GDNS after our notification, but there are still other vulnerable resolvers out there, including OpenDNS. One could only think of the possible damage that can be done if an attacker decide to employ a large botnet to send frequent queries, such as the Mirai botnet [4].

Alternatively, hijacking *only one* popular domain and misconfiguring its NS records would also suffice, as in the case with the anonymous European ccTLD (Figure 11). In this way, it is likely that vulnerable resolvers would be automatically found by the regular stream of user queries.

Once resolvers start looping, the effect on the authoritative servers will depend largely on the attack size versus the authoritative servers's capacity, and there is a large variation among TLDs when it comes to capacity, given there is large variation in the number of authoritative servers and anycast instances per ccTLD.

Most TLDs are likely to suffer at least partial unavailability if faced with 100s of thousands of queries per second. Once down, the consequences can be catastrophic: in case of country-code TLD, most of official services, banks, online shopping and others would become unreachable.

Collateral damage: an attack against a particular TLD may have impact a series of others, given they may share parts of the same infrastructure [3, 15], by using the same authoritative DNS providers. When Dyn DNS was attacked, multiple DNS zones were affected. When some of the Root DNS servers were attack in 2015 [37], parts of the Netherlands' `.nl` ccTLD was also affected [24].

7 CONCLUSIONS

The existence of DNS configuration loops have been previously documented in RFCs. We showed how such loops, combined with vulnerable resolvers, can cause DoS attacks against authoritative servers.

What makes TsuNAME particularly dangerous is that it can be exploited to attack critical DNS infrastructure like large TLDs or ccTLDs, potentially affecting country-specific services and induce collateral damage. We observed 50% traffic increases due to TsuNAME in production in .nz traffic, and 900% more traffic in a EU-based ccTLD – both cases due to configuration errors with *only two* domains, and not real attacks. In controlled experiments we have generated 5600 queries/s using test domains with no query history. An adversary could achieve far more damage using multiple domains and using a large botnet to probe open resolvers besides its own local resolvers.

To prevent TsuNAME to be used for DDoS, we responsible disclosed it to vendors and operators, and Google and OpenDNS promplty fixed their software. We also released CycleHunter, a tool for authoritative server operators to detect cyclic dependencies from their zones, so they can be repaired before attacks occur. We intend to submit an IETF draft recommending resolvers to detect and cache NS loops.

Acknowledgments

We would like to thank Puneet Sood, Warren Kumari, Brian Dickson, Barry Greene, Keith Mitchell, Denesh Bhabuta, Benno Overeinder, Dario Ciccarone, Ralph Weber and the DNS-OARC community for their feedback and support in handling this vulnerability disclosure. We thank the Anonymous European ccTLD operator for kindly sharing their experience and their traffic graph when they experienced a TsuNAME event (Figure 11).

We thank all contributors to CycleHunter on Github: https://github.com/SIDN/CycleHunter/graphs/contributors

We also thank the Ripe Atlas measurement platform [34, 35], which we used extensively in the characterization of this vulnerability in [22].

REFERENCES

[1] 1.1.1.1. 2018. The Internet's Fastest, Privacy-First DNS Resolver. https://1.1.1.1/. https://1.1.1.1/

[2] Gautam Akiwate, Mattijs Jonker, Raffaele Sommese, Ian Foster, Geoffrey M. Voelker, Stefan Savage, and KC Claffy. 2020. Unresolved Issues: Prevalence, Persistence, and Perils of Lame Delegations. In *Proceedings of the ACM Internet Measurement Conference* (Virtual Event, USA) *(IMC '20)*. Association for Computing Machinery, New York, NY, USA, 281–294. https://doi.org/10.1145/3419394.3423623

[3] Mark Allman. 2018. Comments on DNS Robustness. In *Proceedings of the Internet Measurement Conference 2018* (Boston, MA, USA) *(IMC '18)*. Association for Computing Machinery, New York, NY, USA, 84–90. https://doi.org/10.1145/3278532.3278541

[4] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *Proceedings of the 26th USENIX Security Symposium*. USENIX, Vancouver, BC, Canada, 1093–1110. https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-antonakakis.pdf

[5] cert.gov. 2021. Vulnerability Disclosure Policy. https://vuls.cert.org/confluence/display/Wiki/Vulnerability+Disclosure+Policy.

[6] CycleHunter. 2021. GitHub - SIDN/CycleHunter: Python software that reads zone files, extract NS records, and detect cyclic dependencies. https://github.com/SIDN/CycleHunter.

[7] CZ-NIC. 2021. Knot DNS. https://www.knot-dns.cz/

[8] Google. 2020. Public DNS. https://developers.google.com/speed/public-dns/. https://developers.google.com/speed/public-dns/

[9] Google Project Zero. 2021. Vulnerability Disclosure FAQ. https://googleprojectzero.blogspot.com/p/vulnerability-disclosure-faq.html.

[10] P. Hoffman, A. Sullivan, and K. Fujiwara. 2018. *DNS Terminology*. RFC 8499. IETF. http://tools.ietf.org/rfc/rfc8499.txt

[11] ICANN. 2020. Centralized Zone Data Service. https://czds.icann.org/.

[12] Internet Assigned Numbers Authority (IANA). 2020. Root Files. https://www.iana.org/domains/root/files.

[13] Internetstiftelsen. 2020. Zone Data. https://zonedata.iis.se/.

[14] ISC . 2021. BIND 9 . https://www.isc.org/bind/.

[15] Aqsa Kashaf, Vyas Sekar, and Yuvraj Agarwal. 2020. Analyzing Third Party Service Dependencies in Modern Web Services: Have We Learned from the Mirai-Dyn Incident?. In *Proceedings of the ACM Internet Measurement Conference* (Virtual Event, USA) *(IMC '20)*. Association for Computing Machinery, New York, NY, USA, 634–647. https://doi.org/10.1145/3419394.3423664

[16] A. Kumar, J. Postel, C. Neuman, P. Danzig, and S. Miller. 1993. *Common DNS Implementation Errors and Suggested Fixes*. RFC 1536. IETF. http://tools.ietf.org/rfc/rfc1536.txt

[17] M. Larson and P. Barber. 2006. *Observed DNS Resolution Misbehavior*. RFC 4697. IETF. http://tools.ietf.org/rfc/rfc4697.txt

[18] P.V. Mockapetris. 1987. *Domain names - concepts and facilities*. RFC 1034. IETF. http://tools.ietf.org/rfc/rfc1034.txt

[19] P.V. Mockapetris. 1987. *Domain names - implementation and specification*. RFC 1035. IETF. http://tools.ietf.org/rfc/rfc1035.txt

[20] Giovane C. M. Moura. 2021. OARC Members Only Session: Vulnerability Disclosure (DDoS). https://indico.dns-oarc.net/event/37/contributions/821/. https://indico.dns-oarc.net/event/37/contributions/821/

[21] Giovane C. M. Moura, Sebastian Castro, Wes Hardaker, Maarten Wullink, and Cristian Hesselman. 2020. Clouding up the Internet: How Centralized is DNS Traffic Becoming?. In *Proceedings of the ACM Internet Measurement Conference* (Virtual Event, USA) *(IMC '20)*. Association for Computing Machinery, New York, NY, USA, 42–49.

[22] Giovane C. M. Moura, Sebastian Castro, John Heidemann, and Wes Hardaker. 2021. *tsuNAME: exploiting misconfiguration and vulnerability to DDoS DNS*. Technical Report 2021-01. SIDN Labs. https://tsuname.io/tech_report.pdf. https://doi.org/paper.pdf

[23] Giovane C. M. Moura, Sebastian Castro, John Heidemann, and Wes Hardaker. 2021. *tsuNAME: public disclosure and Security Advisory*. Technical Report 2021-05. SIDN Labs, InternetNZ and USC/ISI https://tsuname.io/advisory.pdf.

[24] Giovane C. M. Moura, Ricardo de O. Schmidt, John Heidemann, Wouter B. de Vries, Moritz Müller, Lan Wei, and Christian Hesselman. 2016. Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event. In *Proceedings of the ACM Internet Measurement Conference*. ACM, Santa Monica, California, USA, 255–270. https://doi.org/10.1145/2987443.2987446

[25] Giovane C. M. Moura, John Heidemann, Moritz Müller, Ricardo de O. Schmidt, and Marco Davids. 2018. When the Dike Breaks: Dissecting DNS Defenses During DDoS. In *Proceedings of the ACM Internet Measurement Conference* (johnh: pafile). Boston, MA, USA, 8–21.

https://doi.org/10.1145/3278532.3278534

[26] NL Netlabs. 2021. UNBOUND. https://www.nlnetlabs.nl/projects/unbound/about/.

[27] OpenDNS. 2021. Setup Guide: OpenDNS. https://www.opendns.com/. https://www.opendns.com/

[28] Vasileios Pappas, Zhiguo Xu, Songwu Lu, Daniel Massey, Andreas Terzis, and Lixia Zhang. 2004. Impact of Configuration Errors on DNS Robustness. *SIGCOMM Comput. Commun. Rev.* 34, 4 (Aug. 2004), 319–330. https://doi.org/10.1145/1030194.1015503

[29] Nicole Perlroth. 2016. Hackers Used New Weapons to Disrupt Major Websites Across U.S. *New York Times* (Oct. 22 2016), A1. http://www.nytimes.com/2016/10/22/business/internet-problems-attack.html

[30] PowerDNS. 2021. Changelogs for all pre 4.0 releases. https://doc.powerdns.com/recursor/changelog/pre-4.0.html.

[31] Quad9. 2018. Quad9 | Internet Security & Privacy In a Few Easy Steps. https://quad9.net.

[32] Audrey Randall, Enze Liu, Gautam Akiwate, Ramakrishna Padmanabhan, Geoffrey M. Voelker, Stefan Savage, and Aaron Schulman. 2020. Trufflehunter: Cache Snooping Rare Domains at Large Public DNS Resolvers. In *Proceedings of the ACM Internet Measurement Conference* (Virtual Event, USA) *(IMC '20)*. Association for Computing Machinery, New York, NY, USA, 50–64. https://doi.org/10.1145/3419394.3423640

[33] RIPE NCC. 2021. RIPE Atlas Measurement IDS. https://atlas.ripe.net/measurements/ID. , where ID is the experiment ID: New Domain:25666966, Recurrent:25683316 , One-off-AfterGoogle: 29078085, RecurrentAfterGoogle: 29099244, probe52196:29491104, TripeDep:29559226, CNAME: 29560025.

[34] RIPE NCC Staff. 2015. RIPE Atlas: A Global Internet Measurement Network. *Internet Protocol Journal (IPJ)* 18, 3 (Sep 2015), 2–26.

[35] RIPE Network Coordination Centre. 2020. RIPE Atlas. https://atlas.ripe.net.

[36] RIPE Network Coordination Centre. 2020. RIPE Atlas - Raw data structure documentations,https://atlas.ripe.net/docs/data_struct/.

[37] Root Server Operators. 2015. Events of 2015-11-30. http://root-servers.org/news/events-of-20151130.txt.

[38] Root Server Operators. 2020. Root DNS. http://root-servers.org/.

[39] Root Zone file. 2020. Root. http://www.internic.net/domain/root.zone.

[40] SIDN Labs. 2020. ENTRADA - DNS Big Data Analytics. https://entrada.sidnlabs.nl/.

[41] SIDN Labs. 2021. Tsuname.io. https://tsuname.io.

[42] Raffaele Sommese, Giovane CM Moura, Mattijs Jonker, Roland van Rijswijk-Deij, Alberto Dainotti, Kimberly C Claffy, and Anna Sperotto. 2020. When parents and children disagree: Diving into DNS delegation inconsistency. In *International Conference on Passive and Active Network Measurement*. Springer, 175–189.

[43] Suzanne Goldlust. 2018. Using the Response Rate Limiting Feature. https://kb.isc.org/docs/aa-00994.

[44] S. Thomson, C. Huitema, V. Ksinant, and M. Souissi. 2003. *DNS Extensions to Support IP Version 6*. RFC 3596. IETF. http://tools.ietf.org/rfc/rfc3596.txt

[45] Chris Williams. 2019. Bezos DDoS'd: Amazon Web Services' DNS systems knackered by hours-long cyber-attack. https://www.theregister.co.uk/2019/10/22/aws_dns_ddos/.

[46] Maarten Wullink, Giovane CM Moura, Moritz Müller, and Cristian Hesselman. 2016. ENTRADA: A high-performance network traffic data streaming warehouse. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 913–918.