TsuNAME Public Disclosure and Security Advisory

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

May 6, 2021

Summary

This document describes TsuNAME, a vulnerability that can be exploited to DDoS *authoritative DNS servers*, such as the ones ran by top-level domain (TLDs) operators.

We provide recommendations to authoritative server operators on how to mitigate the issue in their zones (§4) and to resolver software developers and operators to check if their software is vulnerable (§5).

Together with this Security Advisory, we release also a technical report that includes a detailed analysis of TsuNAME events [7].

1 Introduction

We have identified a DNS vulnerability we call "TsuNAME". It affects DNS resolvers and can be exploited to attack authoritative servers. Resolvers vulnerable to TsuNAME will send *non-stop* queries to authoritative servers that have *cyclic dependent records* [10]. While one resolver is unlikely to overwhelm an authoritative server, the aggregated effect from *many* looping, vulnerable recursive resolvers may as well do.

This document is divided as follows:

- §2 presents the vulnerability
- §3 discusses the potential impact on authoritative servers
- §4 shows how authoritative server operators can remediate their zones
- §5 shows how resolver operators can detect if their software is bogus.
- §6 covers the responsible disclosure steps we have taken since then, and shows two cases of large public resolver operators that fixed their software.

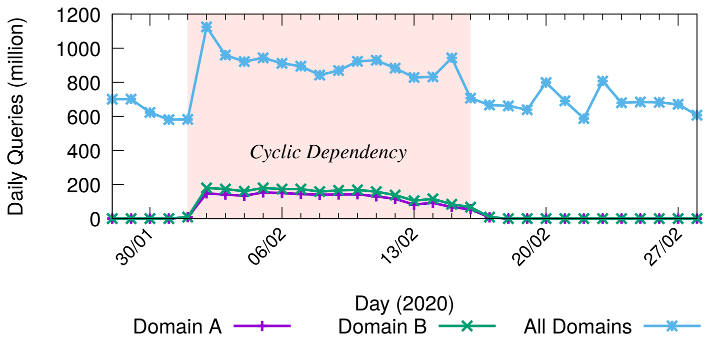

Figure 1: Query volume timeseries for all domains and cyclic dependent domains (A and B)

2 The TsuNAME vulnerability

The TsuNAME vulnerability allows for an adversary to exploit vulnerable recursive resolvers, which will then send a *very large* volume of queries to the targeted authoritative servers.

For this to happen, three things are necessary:

1. Cyclic dependent NS records (§2.2)
2. Vulnerable recursive resolvers (§2.3)
3. User queries to start/drive the process

This is not theoretical, and has happened multiple times in the past – we have evidence of happening at least with four ccTLDs and one gTLD. Next we described what happened to the `.nz` ccTLD.

2.1 The `.nz` event

On 2020-02-01, the authoritative servers for the `.nz` experienced a *surge* in its total traffic – roughly a 50% increase, from 800M to 1.2B daily queries (Figure 1). Upon investigation, the `.nz` operators determined that the cause of this surge in traffic was due to a *configuration error* in *two domains only*. The error was that the two domains were misconfigured with cyclic dependencies (§2.2).

This configuration error caused the daily queries for these two domains to go from 35k to an average of 269M (peaking at 334M in one day) – a staggering 7643× traffic growth (shaded area in Figure 1, as seen by the `.nz` authoritative servers). The event only stopped after 16 days when `.nz` operators removing the cyclic dependency by removing the affected delegation.

Notice that a simple misconfiguration of two domains lead to 50% traffic growth. One may wonder what would happen if a motivated attack would carry out this with hundreds or thousands of domains.

2.2 Cyclic Dependency

TsuNAME can only occur in the presence of *cyclic dependency* [10], which is a NS configuration error that occurs when NS records for two zones point to each other. It is very easy for a registrant to perform that for hundreds or even thousands of domains in TLDs with open registrations, such as `.org`.

Let's use as examples two zones with cyclically dependent records in the rest of this document. Listing 1 shows a part of zone file of `essedarius.net`. Lines #1 and #2 define the authoritative servers for the zone.

Listing 1: DNS Zone file: `essedarius.net`

```
1  essedarius.net.        1        IN       NS       ns1.example.nl.
2  essedarius.net.        1        IN       NS       ns2.example.nl.
```

If a user queries for any record under `essedarius.net`, then the DNS resolver will have to query the authoritative server `example.nl`.

Listing 2 shows the zone file of `example.nl`. We see that `example.nl` has as authoritative servers `ns[3,4].essedarius.net`. In this case `essedarius.net.` is *cyclically dependent* with `example.nl.` (`essedarius.net` $\leftrightarrow$ `example.nl`), given they point to each other, and there is no way for a resolver to retrieve A and AAAA records of these cyclically dependent NS records. Consequently, these domains *cannot be resolved*.

Listing 2: DNS Zone file: `example.nl`

```
1  example.nl.        1        IN       NS       ns3.essedarius.net.
2  example.nl.        1        IN       NS       ns4.essedarius.net.
```

Note that, in the previous example, there is no way for an operator of a single zone to know if a domain is cyclic dependent or not by simply doing static analysis, *i.e.*, only looking its own zone files. It is necessary to *actively* query the domain to determine that, in order to have a view of other zones. For that end, we have developed `CycleHunter`, an open-source tool that can be used to detect cyclic dependencies (§5).

2.3 Vulnerable Recursive Resolvers

We refer to *vulnerable recursive resolvers* as resolvers that when encountering cyclically dependent NS records in their recursive resolution process, they do at least one of the two (or both):

1. Do not detect the cycle, and start looping non-stop between cyclically dependent NS records.

2. Do not cache cyclically dependent NS records.

Loop as source of traffic growth: Suppose an user asks a vulnerable recursive resolver for the A record of `essedarius.net` . This resolver will ask one of the authoritative servers of `essedarius.net` (lines # 1 and 2 in Listing 1) for this record, and will be answered with lines # 1 and # 2.

That NS response will bring the resolver the authoritative servers of `example.nl` (lines #1 and #2 in Listing 2), which will also answer with a NS record that points back to `essedarius.net`.

If the resolver, at this stage, is oblivious to the cycle, it will simply bounce back from zone to zone, sending non-stop queries to the authoritative servers of both parent zones (lines #1 and #2 in both listings). These are the servers, which, in turn, *will experience traffic growth*. That is exactly what happened with `.nz`, as shown in Figure 1.

In other cases, the vulnerable recursive resolver will time out after making some number of queries. In this case, while there will not be endless traffic, there will be an amplification factor that increases traffic generated by the initial query. (If each name times out after ten attempts and there is a loop of two names, it may cause a 20× repetition in a single query from a stub resover.)

In our experiments, we have seen resolvers looping non-stop for hours. Please refer to §3.1 and §5 in [7] for more details.

Giving there may be multiple layers of resolvers between a client and an authoritative server, it is important for the last resolver in the chain to *cache* cyclic dependent records, so in the advent of a looping forwarder querying the last resolver, non-stop, that would limit the amount of queries that are passed to the authoritative servers.

2.4 User Queries

Queries by recursive resolvers are triggered from queries from stub resolvers, which are in turn triggered by user running an application. The user, application, and stub resolver can each cause retries that amplify the impact of a cycle.

Large events like the 2020 `.nz` event are likely a combination of newly injected external queries combined with amplification from vulnerable recursive resolvers. Although we have reproduced queries that terminate after amplification, and queries that cycle repeatedly, we suspect that the 2020 `.nz` event reflects some degree of new external queries.

3 Impact on authoritative servers

Once vulnerable recursive resolvers encounter cyclic dependent records, they will begin to loop. However, what authoritative servers receive this traffic, that can ultimately become a DDoS?

It is the *parent zone authoritative servers*. In our example in Listing 1 and Listing 2, it will be the `.nl` and `.net` authoritative servers that will receive the extra queries, given the resolvers cannot pass from this level in the recursive resolution of domains.

For a authoritative server operator, such as TLD operator, the doom's day scenario would involve registrants changing their default NS records to cyclic dependent ones, especially if this is done with hundreds or thousand domains at once.

We have seen in Figure 1 that the misconfiguration of 2 domains led to a 50% traffic growth on `.nz`. But it was not the only instance.

After private disclosures (§6), we have been contacted by an anonymous European ccTLD experienced 10x traffic growth when also two domains were misconfigured with cyclic dependencies.

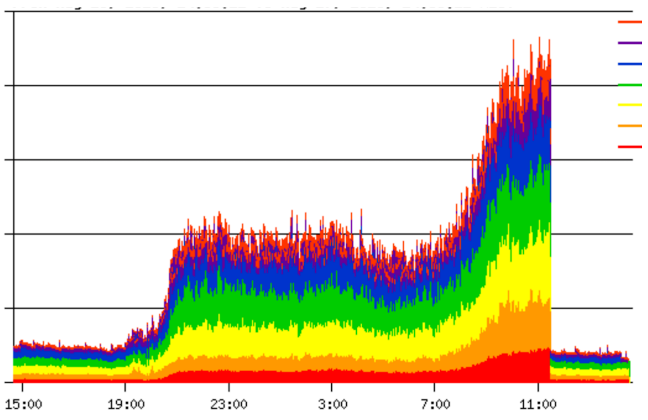

Figure 2: TsuNAME event at an Anonymous EU-based ccTLD operator.

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 2 shows a time-series of queries (y axis anonymized by the ccTLD 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 after fixing the records.

4 Recommendation for authoritative server operators

Authoritative servers are the *target* of the TsuNAME vulnerability. In the case of Listing 1 and Listing 2, the servers listed in both lines #1 and #2 would be the ones seeing a surge in traffic, *i.e*,, the *parent zone* authoritative servers of the `.nl` and `.net` zones.

Once resolvers start looping, *i.e.*, sending non-stop queries for cyclic dependent domain, which may overwhelm their parent authoritative servers.

To mitigate the risks of TsuNAME-related attacks, we recommend authoritative sever operators to *detect* and *remove* cyclic dependencies from their zones. To that end, we provide `CycleHunter`, an open-source tool that reads zone files and scrutinize NS records searching for cyclic dependencies. `CycleHunter` can be downloaded at `https://github.com/SIDN/CycleHunter`.

Please note that given that NS records can change at *any* time, there is no *permanent* solution. In other words, if a DNS zone has no cyclically dependent NS records at time t, it means that this zone is not vulnerable at only that particular time t. We therefore also recommend that registrars run `CycleHunter` on a regular basis, for instance as part of their domain name registration process.

5 Recommendations for resolver operators

To mitigate the traffic surge from resolvers to authoritative servers caused by the TsuNAME vulnerability, resolver operators should guarantee that their resolvers (i) do not loop in the presence of cyclic dependencies and (ii) cache the results of cyclic dependent records.

For example, in the Listing 1 and Listing 2 examples, that would involve in detecting that these NSes are *unresolvable*, and caching them – possibly as SERVFAIL [6]. Then, any subsequent queries to these delegations will notice that there is no resolvable NS record for this zone, and will be answered as SERVFAIL from the cache, reducing the volume of queries to authoritative servers.

You can test your resolver by setting up *third-level* cyclic dependent zones and sending queries to your resolver, to determine if starts looping and if cache the results. We include two tests in Appendix A that you can do with your own DNS resolvers.

5.1 Evaluated resolvers

We configure a test zone with cyclic dependency and evaluate popular DNS resolvers: Unbound (v 1.6.7) [8], BIND (v 9.11.3) [5], and KnotDNS (v 5.1.3) [1], on a VM on AWS EC2 (FRA). We found that none of the start looping in the presence of cyclic dependent domains, hence are not vulnerable to it.

6 Responsible disclosure and mitigation

We wish to protect authoritative server operators from TsuNAME based attacks. To do that, we have carefully coordinated responsible disclosure to vulnerable parties, at least 90 days *prior* to this public disclosure.

The first party we notified was Google, given Google Public DNS (GDNS) [4] was responsible for most queries during the `.nz` event. They have fixed the issue since then. Another party we notified was Cisco OpenDNS [9], which has also fixed fixed their software too.

Table 1 shows the timeline of notification about TsuNAME. We first choose to notify first developers and folks that run most vulnerable recucrisve resolvers, such as Google. Then, we carried out a private notification section for DNS-OARC members only, during OARC 34 [2]. This private notification, among trusted partners, allowed the community to contribute and provide feedback, for example, by significantly improving `CycleHunter`. We are thankful to all of them.

Acknowledgments

We would like to thank Puneet Sood, Warren Kumari, Brian Dickson, Barry Greene, Keith Mitchell, 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 2).

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 1: TsuNAME disclosure timeline
We will also public disclose TsuNAME during the next OARC 35 [3].

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

We also thank the Ripe Atlas measurement platform [12, 11], which we used extensively in the characterization of this vulnerability in [7].

Legal Disclaimer

Appendix

A Evaluating your resolvers

To test your resolver software, set up cyclically dependent delegations, as shown in Listing 1 and Listing 2. We **strongly recommend creating third-level domain names** instead of second-level (*e.g.*, `example.nl`) given that cyclically dependent second-level domains will stress authoritative servers of their respective TLDs (§4).

After creating these cyclically dependent delegations, we suggest the following tests:

A.1 Test 1: Loop Detection

1. Clean the cache of your resolver
2. Monitor the traffic between the resolver and the Internet

3. Send ONE query to your for a domain under the misconfigured delegation.

    - Compute how many queries are then sent to the parent authoritative servers of both misconfigured zones (lines #1 and #2 of Listing 1 and Listing 2)
    - Determine if your resolver loops indefinitely, or if eventually stop sending queries to the authoritative servers. You may need to monitor for various minutes or hours.

Please notice that the resolver may send a `SERVFAIL` response to your client, but it may remain looping, sending non-stop queries to the authoritative servers.

For a reference, you may want to check `Unbound`'s source code, which includes various types of cycle/loop detection, as described in their changelog[1].

A.2 Test 2: Caching Cyclic Records and Amplification

1. Clean the cache of your resolver
2. Monitor the traffic between the resolver and the Internet
3. Send ONE query to your for a domain under the misconfigured delegation.
4. Then, send this query multiple times every 5 s (or other short interval)

    - Compute how many queries are then sent to the parent authoritative servers of both misconfigured zones (lines #1 and #2 of Listing 1 and Listing 2)
    - Determine if the new, recurrent queries from your client (`dig` in this case) cause your resolver to send many more queries to reach the authoritative servers, or if they are answered from cache.

If new user queries (`dig`) lead to more queries to the authoritative servers, you resolver is then vulnerable to TsuNAME.

References

[1] CZ-NIC. Knot dns, January 2021.

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

[3] Giovane Moura. Public Disclosure DNS vulnerability. https://indico.dns-oarc.net/event/38/contributions/849/, May 2021.

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

[1] https://github.com/NLnetLabs/unbound/blob/master/doc/Changelog

[5] ISC . BIND 9 . https://www.isc.org/bind/, January 2021.

[6] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034, IETF, November 1987.

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

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

[9] OpenDNS. Setup Guide: OpenDNS. https://www.opendns.com/, March 2021.

[10] Vasileios Pappas, Zhiguo Xu, Songwu Lu, Daniel Massey, Andreas Terzis, and Lixia Zhang. Impact of configuration errors on DNS robustness. *SIGCOMM Comput. Commun. Rev.*, 34(4):319–330, August 2004.

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

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