Silence is not Golden: Disrupting the Load Balancing of Authoritative DNS Servers

Fenglu Zhang
Tsinghua University
Beijing, China
zfl23@mails.tsinghua.edu.cn

Baojun Liu*
Tsinghua University
Beijing, China
lbj@tsinghua.edu.cn

Eihal Alowaisheq
King Saud University
Riyadh, Saudi Arabia
ealowaisheq@ksu.edu.sa

Jianjun Chen
Tsinghua University; Zhongguancun Laboratory
Beijing, China
jianjun@tsinghua.edu.cn

Chaoyi Lu
Tsinghua University
Beijing, China
luchaoyi@tsinghua.edu.cn

Linjian Song
Alibaba Group
Beijing, China
linjian.slj@alibaba-inc.com

Yong Ma
Alibaba Group
Beijing, China
mayong.my@alibaba-inc.com

Ying Liu*
Tsinghua University
Beijing, China
liuying@cernet.edu.cn

Haixin Duan
Tsinghua University; Quancheng Laboratory
Beijing, China
duanhx@tsinghua.edu.cn

Min Yang
Fudan University
Shanghai, China
m_yang@fudan.edu.cn

ABSTRACT

Authoritative nameservers are delegated to provide the final resource record. Since the security and robustness of DNS are critical to the general operation of the Internet, domain owners are required to deploy multiple candidate nameservers for load balancing. Once the load balancing mechanism is compromised, an adversary can manipulate a large number of legitimate DNS requests to a specified candidate nameserver. As a result, it may bypass the defense mechanisms used to filter malicious traffic that can overload the victim nameserver, or lower the bar for DNS traffic hijacking and cache poisoning attacks.

In this study, we report on a class of DNS vulnerabilities and present a novel attack, named *Disablance*, that targets the domains with different NS records severing to multiple sites of authoritative servers. The attack is made possible by a misconfiguration of nameservers that ignores domains outside their authority, combined with recursive resolvers that use a globally shared status for nameserver selection. By targeting authoritative nameservers configured by a large number of domains, Disablance allows adversaries to *stealthily sabotage the DNS load balancing for authoritative nameservers at a low cost*. Through simply configuring the DNS records for a domain under their control to point to the targeted nameservers and performing a handful of requests, adversaries can temporarily manipulate a given DNS resolver to overload a specific authoritative server. Therefore, Disablance can redirect benign DNS requests for *all* hosted domains to the specific nameserver and disrupts the load balancing mechanism. Our extensive study proves the security threat of Disablance is *realistic* and *prevalent*. First, we demonstrated that mainstream DNS implementations, including BIND9, PowerDNS, and Microsoft DNS, are vulnerable to Disablance. Second, we developed a measurement framework to measure vulnerable authoritative servers in the wild. 22.24% of top 1M FQDNs and 3.94% of top 1M SLDs were proven can be the victims of Disablance. Our measurement results also show that 37.88% of stable open resolvers and 10 of 14 popular public DNS services can be exploited to conduct Disablance, including Cloudflare and Quad9. Furthermore, the critical security threats of Disablance were observed and acknowledged through in-depth discussion with a world-leading DNS service provider. We have reported discovered vulnerabilities and provided recommendations to the affected vendors. Until now, Tencent Cloud (DNSPod) and Amazon have taken action to fix this issue according to our suggestions.

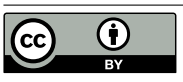
CCS CONCEPTS

• **Security and privacy** → **Network security**; • **Networks** → *Naming and addressing*.

KEYWORDS

Domain name system; DNS security; Nameservers; Load balancing

*Both authors are corresponding authors.



CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11.
https://doi.org/10.1145/3576915.3616647

ACM Reference Format:
Fenglu Zhang, Baojun Liu, Eihal Alowaisheq, Jianjun Chen, Chaoyi Lu, Linjian Song, Yong Ma, Ying Liu, Haixin Duan, and Min Yang. 2023. Silence is not Golden: Disrupting the Load Balancing of Authoritative DNS Servers. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3616647

1 INTRODUCTION

The Domain Name System (DNS) provides critical services for mapping user-friendly domains to IP addresses. Nowadays, DNS serves as one of the fundamental cornerstones of mainstream Internet services and provides trusted anchors in modern security paradigms, such as digital certificate issuance [28, 36] and email sender authentication [43, 45]. Therefore, the *robustness of domain naming services* is crucial to global Internet activities.

To improve the stability of naming services and resist denial-of-service (DoS) attacks, a *"built-in" load balancing* mechanism was introduced by DNS specifications [7]. Since the DNS is structured as a distributed system, the delegation of a domain can depend on a set of authoritative nameservers to facilitate robustness. As the DNS specification requires, each DNS zone should maintain at least two authoritative servers [62]. RFC 2182 further requires these nameservers to be geographically and topologically diverse [72]. As such, domain owners often deploy multiple authoritative servers. The percentage of SLDs that meet the requirements of the above standards is increasing over time [7].

The consequences of compromising the DNS load balancing mechanism of authoritative servers are extremely severe. Unlike most classical DoS attacks, disrupting load balancing allows an adversary to redirect legitimate DNS traffic and overload the victim without generating malicious traffic. This fact has been demonstrated in a body of work. For example, such overload can bypass the defense mechanisms that detect and filter malicious traffic [34, 61, 96]. In addition, when adversaries disable DNS load balancing, they significantly lower the bar of carrying out other attacks, such as traffic hijacking and cache poisoning, since most traffic is directed to the specific nameserver. For instance, Dai et al. [26] confirmed that eliminating candidates provided by DNS load balancing is essential for obtaining fraudulent certificates from Let's Encrypt. Furthermore, such attacks provide an opportunity for adversaries to manipulate how the recursive resolver distributes DNS traffic among nameservers. As a result, the impact of such attacks is more severe on the infrastructure of DNS-based load balancing systems, such as upper-layer web applications.

Research gap. Given the significance, though, few studies have investigated the security risks of the standard solution for DNS load balancing. A recent study presented a taxonomy of attacks on DNS load balancing [26], including *IP fragmentation*, *rate-limiting*, and *low-rate bursts*. However, we believe that the real-world impact of the above attacks are strictly limited [54, 56, 99, 100]. As evidence, less than 0.7% of authoritative servers of Alexa Top 100K domains are vulnerable to IP fragmentation attacks [99]. Rate-limiting attacks rely on brute-force DoS traffic, which can be easily detected by the targets [56]. And low-rate bursts attack requires accurate time synchronization and is challenging to implement in the real world [100].

Our study. In this paper, we report on a new class of DNS vulnerabilities that *can disrupt the DNS load balancing mechanism of popular domains in a stealthy manner*. These domains are typically configured with multiple `NS` records that point to authoritative servers, which in turn also host many other domains. Especially, our proposed attack can be initiated by performing a small number of DNS queries. Thus, it allows an adversary to break the DNS load balancing mechanism in a cost-effective and stealthy manner.

The root causes of the vulnerabilities we identified are twofold. First, a prevalent misconfiguration of authoritative servers is revealed. Probably due to the consideration of mitigating DoS attacks, extensive authoritative servers are configured to not respond to DNS requests which are *outside of their authority*. Second, the function of selecting candidate nameservers implemented in mainstream recursive resolvers has proven flawed. In general, DNS resolvers attempt to select the authoritative servers with the lowest network latency. If this query fails, it will fail over and attempt the query on the next authoritative server [3, 64, 95, 97]. To improve the efficiency of the following DNS resolution, the *status* of nameservers that *fail to respond* will be cached [11]. Unfortunately, the status could be a *globally shared resource*, in which case it applies to all queries to that specific nameserver.

By combining the above characteristics, adversaries can manipulate the nameserver selection results of recursive resolvers. Specifically, they can register a domain and set up the nameserver records to all of the victim's authoritative servers, except the ones that they wish to overload. The exploitable authoritative servers ignore DNS requests for domains not hosted on them. Then, the attacker generates DNS queries to globally distributed resolvers, including public DNS and open resolvers. Since most of the victim's nameservers failed to respond, the *unavailable* status will be cached at global resolvers. Consequently, the following legitimate queries for *all domains* delegated to such nameservers will be forwarded to the targeted authoritative server. We refer to the proposed attack as DNS Load Balancing Disabler (**Disablance**).

Our findings. In this paper, we performed a systematic evaluation study for Disablance and demonstrated that the proposed attack is *realistic* and *efficient* in the real world. First, through reviewing of source code and conducting simulation experiments, we confirmed popular DNS software (BIND9 [38], PowerDNS [68] and Microsoft DNS [58, 59]) were vulnerable and can be leveraged to initiate Disablance. It should be highlighted that an adversary can cause a DNS resolver to overload a given authoritative server with thousands of DNS queries by sending several crafted DNS requests.

Second, we designed and implemented a measurement framework named DMiner to understand the pervasiveness of affected authoritative servers. Our measurement results show that, **22.24%** of the top 1M SecRank [93] FQDNs and **3.94%** of the top 1M Tranco [48] SLDs can be targeted by Disablance. Some exploitable popular domains are serving critical network services, such as E-commerce, cloud, and medical services. Even some world-leading DNS hosting service providers are proven to be affected, such as Tencent Cloud (DNSPod) [82] (hosting 6.26% of the top 1M SecRank and 0.81% of the top 1M Tranco). To roughly estimate the traffic adversaries may impact, we analyzed the passive DNS data of FQDNs corresponding to the vulnerable SLDs we identified. During the one-week period that DMiner tested vulnerable nameservers, we

observed **61,601,477** unique FQDNs, indicating that a large amount of DNS traffic can be diverted by Disablance.

Third, Disablance can amplify and divert legitimate requests for **all** hosted domains to a given authoritative server. Zooming in a single affected domain, Disablance amplifies its legitimate DNS traffic by an average of 8.51 times (46 times maximum). Interestingly, the domains with high requirements for availability and stability, which deploy more authoritative servers for DNS load balancing, are suffering from greater amplification impact (e.g., 32 times for eight financial institutions and 46 times for a technology company).

We reported the identified vulnerabilities and had an in-depth discussion with leading DNS service providers. Surprisingly, a DNS provider reported evidence of Disablance in their network. Over 81 hours, the public recursive resolvers they operated on were observed to query an oversea authoritative server without referring any candidates closer. Extremely higher network latency (21 times more than normal) for the targeted nameserver was also reported. They acknowledged the real-world security threats of Disablance against their hosting services, including bypassing existing DoS defense mechanisms.

Contributions. The contributions of this paper are listed below:

- *Novel attack.* We conducted an in-depth study on the DNS load balancing mechanism and uncovered a vulnerability that adversaries can exploit to disrupt its functionality.
- *Comprehensive measurement to evaluate the attacks.* We developed a measurement framework and systematically evaluated the real-world impact of our proposed attack.
- *Responsible disclosure.* We have responsibly disclosed issues to vendors and service providers with mitigation options. Until now, Tencent Cloud and Amazon have taken action to fix it.

2 BACKGROUND

2.1 DNS Basics

The resolution process starts when a client requests a recursive resolver to resolve a domain. Since the domain name space is organized as a hierarchical structure, the recursive resolver should forward the query to a root server. Subsequently, the resolver will contact the top-level domain (TLD) server, the second-level domain (SLD) server, and the domain's authoritative server. Once the recursive resolver receives the final answer from the authoritative server, it forwards the response to the client [62].

Recursive resolvers and authoritative servers play crucial roles in the DNS resolution process. Recursive resolvers provide DNS resolution for their clients. Some recursive resolvers are open and provide resolution services for anyone connected to the Internet (e.g., Google [32], Quad9 [75], OpenDNS [20], etc.), while others only serve a specific network (e.g., ISPs, university campus, etc.). Authoritative servers are designed to be authorized to provide resource records for a specific domain, and should not provide answers for other domains which are outside their authority.

2.2 DNS Hosting Service

Overview of DNS hosting. DNS hosting is a third-party network service that provides DNS resolution services. DNS hosting platform simplifies the complex work for customers and helps manage DNS resource records efficiently [98]. Usually, DNS hosting services are integrated with domain registrars or web hosting providers. A number of well-known DNS hosting services include Cloudflare [22], Dyn [70], Tencent Cloud [82], Alibaba Cloud [6], and Namecheap [66]. Third-party DNS hosting services are widely adopted by popular domains and have even contributed to the centralization of Internet traffic. Recent researches show that 89% of the top 100K websites use third-party DNS services [40, 63].

Load balancing for authoritative servers. To achieve the goals of providing fault tolerance and improving resilience against DoS attacks, operators of DNS hosting services strive to adopt DNS traffic load balancing mechanisms. Specifically, the customer's domain name is usually configured with a set of authoritative servers to handle DNS requests. Moreover, each nameserver can point to several physical instances. As a result, DNS requests toward a hosted domain will be distributed among multiple instances with different IP addresses. When a DNS service provider has multiple servers in different geographical locations, it will provide resilience to worldwide customers. As an example, a recursive resolver witnesses a total of 14 choices (authoritative servers) to query a domain hosted on the Tencent Cloud (DNSPod), as shown in Figure 1.

```
;; ANSWER SECTION:
foo_dnspod.com.      86400    IN    NS    f1g1ns1.dnspod.net.
foo_dnspod.com.      86400    IN    NS    f1g1ns2.dnspod.net.
;; ADDITIONAL SECTION
f1g1ns1.dnspod.net.  172800   IN    A     1.12.0.4
      ...
f1g1ns2.dnspod.net.  172800   IN    A     223.166.151.21
      ...
(15 IP addresses in total)
```

Figure 1: Delegation records for a domain hosting on DNSPod

2.3 Authoritative Servers Selection Strategy

As mentioned above, large DNS services tend to map resource records to a set of physical sites. However, a resolver is only required to select and interact with one physical site during a resolution process. The DNS specification describes that: A recursive resolver should "*find the best servers to ask*" [62], and mainstream DNS software implements different selection strategies for this recommendation, such as searching for low latency, picking a nameserver at random or round robin [3, 95]. In 2017, Müller et al. [64] conducted a large-scale re-evaluation study to examine how recursive resolvers select authoritative servers. They demonstrated that most recursive resolvers check all authoritative servers over time, with about *half of the resolvers showing a preference for low latency.*

The strategy of authoritative server selection plays a significant role in the load balancing of authoritative servers. While worldwide resolvers pick the fastest candidate from their perspectives, they are also scheduled to query the geographically nearest authoritative servers. In addition, an overloaded server typically responds with high latency or even fails to respond. This may cause resolvers to consider other idle candidates instead of overloaded ones.

In this paper, we present that an adversary can exploit the selection strategy to restrict the options of responsive nameservers for the resolver, then force into querying a candidate specified by the attacker. As a result, the load balancing of authoritative servers will be invalid.

2.4 Security Implications of Disrupting DNS Load Balancing

Bypassing DoS defense mechanisms and overloading nameservers. Disabling DNS load balancing allows attackers to redirect legitimate DNS traffic to a specified target. Such overloads can bypass various defense mechanisms against traditional DoS attacks, including detecting and filtering malicious traffic [34, 61, 96]. Specifically, detecting mechanisms identify malicious packets according to a series of characterizations. Examples include IP traceback schemes [49] and IP spoof detection [53]. Filtering mechanisms utilize results from detecting mechanisms to filter out the attack streams completely or impose rate limits [61]. However, attackers, who disrupt DNS load balancing mechanisms, divert all legitimate traffic to a victim instead of generating distinguishable malicious traffic. As a result, no malicious traffic or client can be detected or filtered. The choices remaining for victims are increasing hardware capability or ignoring packets from benign clients.

Lowering the bar for traffic hijacking and cache poisoning. To manipulate DNS responses, attackers may hijack the network path of DNS traffic or poison the DNS cache of recursive resolvers. However, DNS load balancing mechanism provides multiple candidate authoritative servers, which can be deployed on different geolocations or autonomous systems (AS). To ensure the success of a hijacking attack, an adversary needs to hijack enough network paths toward all candidate nameservers. Also, the attacker must inject the forged response matching the destination address of candidate nameserver selected by the resolver. Disrupting the load balancing eliminates the possibility for clients to query diverse nameservers at the `NS` record level and at the IP address level. As a result, manipulation of DNS responses becomes less challenging since a unique path is dedicated to victims. In a recent study, researchers proved that disrupting load balancing is essential for advanced attacks, such as acquiring fraudulent TLS certificates [26].

Disrupting the infrastructure of DNS-based load balancing systems. Upper-layer application (e.g., cloud services and API for IoT devices) servers tend to utilize load balancing mechanisms to improve their robustness. One main-stream implementation is DNS-based load balancing [23, 41]. Typically, DNS-based load balancing techniques are deployed on authoritative servers. Some implementations even directly configure each authoritative server to respond with different resource record sets. Therefore, an attack against such implementations can have a damaging impact on the whole infrastructure and the application servers deployed on it (Section 5.4).

3 DISABLANCE ATTACK

In this section, we first introduce the threat model of the proposed attack (Section 3.1). Then, we dive deeper into the attack details (Section 3.2).

3.1 Threat Model

Disablance (DNS Load Balancing Disabler) aims to disrupt the load balancing mechanism of authoritative servers. The key idea is that, by exploiting the response strategy of authoritative servers, an adversary can manipulate the priority of authoritative servers from the view of a resolver and further force the resolver only to select a given authoritative server for future queries.

In this paper, we assume that adversaries have limited capabilities: First, adversaries are off-path. They cannot hijack or eavesdrop on network traffic between clients and servers. Second, adversaries are only required to generate simple DNS queries (i.e., `A` records of a domain). They do not need to craft unusual or malformed packets (e.g., IP address spoofing or triggering IP fragmentation). Third, adversaries are excepted to send packets at a low speed. They can not overload the bandwidth and further trigger the rate limit of any servers.

For the victims, we assume that a set of victim domains utilize authoritative servers provided by DNS hosting services [40]. Considering a recursive resolver shares the *priority status* of nameservers across all domains, adversaries promote the recursive resolver to perform DNS queries targeting a set of authoritative servers, which ignore requests for domains outside their authority. As a result, the resolver will temporarily lower the priority of such authoritative servers. *Following, no matter what domain is being queried, the recursive resolver will avoid selecting the set of authoritative servers for a period* (Section 4). Consequently, this leaves the resolver with a single authoritative server to interact with, which undermines the load balancing mechanism of all the domains hosted on these servers.

Later we will show that Disablance is highly efficient (Section 4 and 5): By just sending several DNS packets, an adversary can make the resolver overload a targeted authoritative server with thousands of queries from legitimate Internet users.

3.2 Attack Overview

As mentioned above, due to the considerations of mitigating DoS traffic and filtering unwanted DNS queries, some authoritative servers are configured to *silently drop requests for domains that are outside of their authority* instead of directly returning any negative answer (e.g., `REFUSED` or `NXDOMAIN`). As we will show in Section 7, operators silently ignore such queries as a defense mechanism.

To illustrate the steps of Disablance, we provide an example of DNS delegation records for a domain, e.g., hostedDomain.com. As shown in Figure 2: the DNS queries for the domain are distributed among different instances (i.e., $IP_{1\text{-}4}$).

```
$ dig hostedDomain.com NS
        ...
;; ANSWER SECTION:
hostedDomain.com. 3600  IN NS ns1.hostingService.com.
hostedDomain.com. 3600  IN NS ns2.hostingService.com.

;; ADDITIONAL SECTION
ns1.hostingService.com.       3600  IN A  IP1
ns1.hostingService.com.       3600  IN A  IP2
ns2.hostingService.com.       3600  IN A  IP3
ns2.hostingService.com.       3600  IN A  IP4
```

Figure 2: DNS records of a hosted domain.

To break the DNS load balancing mechanism, attackers need to generate several unresponsive DNS queries to a given recursive resolver. They first configure the DNS records of an *arbitrary domain* controlled by them to point to authoritative servers of the hosting provider. Note that, *attackers are not required to sign up with the hosting provider.* Then, attackers set the nameserver records to point to all of the provider's authoritative servers except one. We

```
$ dig attack-1.com NS
        ...
;; ANSWER SECTION:
attack-1.com.  3600  IN NS ns2.hostingService.com.

;; ADDITIONAL SECTION
ns2.hostingService.com.       3600  IN A  IP3
ns2.hostingService.com.       3600  IN A  IP4
```

```
$ dig attack-2.com NS
        ...
;; ANSWER SECTION:
attack-2.com.  3600  IN NS ns.attacker.com.

;; ADDITIONAL SECTION
ns.attacker.com.              3600  IN A  IP2
ns.attacker.com.              3600  IN A  IP3
ns.attacker.com.              3600  IN A  IP4
```

Figure 3: DNS records of attacking domains.

refer to this subset as *unresposiveNS*, while the remaining server is *targetedNS*.

Next, they send several DNS queries to the targeted resolver. There are several ways to execute this. For open DNS resolvers, attackers only need to query the resolver directly. For a resolver that only serves a limited network, they can take advantage of large-scale measurement platforms or residential proxies, in which their vantage points are distributed worldwide [52, 80, 92]. Alternatively, they can also spread ads or emails containing content pointing to their domains. Machines in different networks will query attackers' domains while loading such ads or emails [16, 19, 51]. Since no voluntary participation of normal users is required, the popularity of attackers' domain does not affect the influence of Disablance.

Finally, *unresposiveNS* servers silently ignore DNS queries crafted by adversaries. Then, the globally distributed recursive resolvers treat these *unresposiveNS* as unavailable and lower their priority in subsequent DNS resolutions. **All domain names hosted on the hostingService.com** (not just hostedDomain.com) are affected. A huge volume of DNS requests from legitimate users for those domains will overwhelm the *targetedNS* server for a period.

3.3 Attack Details

We present that the Disablance has two technical variants: adversaries divert DNS traffic from legitimate users to a single NS record (DisablanceNS) or a single IP address (DisablanceAddress). Besides, Disablance will severely impact upper-layer applications deployed on DNS-based load balancing systems.

Variant I: DisablanceNS. is a type of Disablance that directs the traffic to an NS record. Adversaries first configure the NS records of their domain to point to a subset of the authoritative servers (i.e., *unresposiveNS*). Then, they query the recursive resolver for the domain. The authoritative servers ignore the query since the domain is not actually hosted in them. As a result, the resolver lowers the priority of servers in *unresposiveNS* and stops querying them for a period. Consequently, the resolver will divert all legitimate queries for domains hosted on the hosting provider to *targetedNS*. We will evaluate the effectiveness of Disablance in Section 4).

An illustration is shown in Figure 4(a). Assuming that the DNS records of hosted domains are configured as specified in Figure 2, and adversaries aim to overload ns1.hostingService.com, the steps of DisablanceNS are as follows: First, adversaries control a domain (e.g., attack-1.com) and configure the NS records of the domain to point to ns2.hostingService.com (attack-1.com in Figure 3). Then, they send several queries for attack-1.com to the recursive resolver. To prevent the resolver from caching their domain, adversaries add a nonce value in each DNS query ([nonce].attack-1.com). The resolver then forwards the query to ns2.hostingService.com. However, authoritative servers ignore this query because attack-1.com is not hosted on them. The recursive resolver then lowers the priority of ns2.hostingService.com and avoids selecting it, causing ns1.hostingService.com to be overwhelmed by legitimate queries. Benign traffic for all domains hosted on targeted authoritative servers can be affected.

Variant II: DisablanceAddress. is a type of Disablance that directs the traffic to a single IP address. Unlike DisablanceNS, the resolver in DisablanceAddress disturbs the priority of authoritative servers based on the responsiveness of IP addresses. Thus, adversaries set the A records for the nameservers of their domain to point to several IP addresses of authoritative servers. We assume that the DNS records of the hosted domains are configured as shown in Figure 2, and adversaries specify IP_1 as the target.

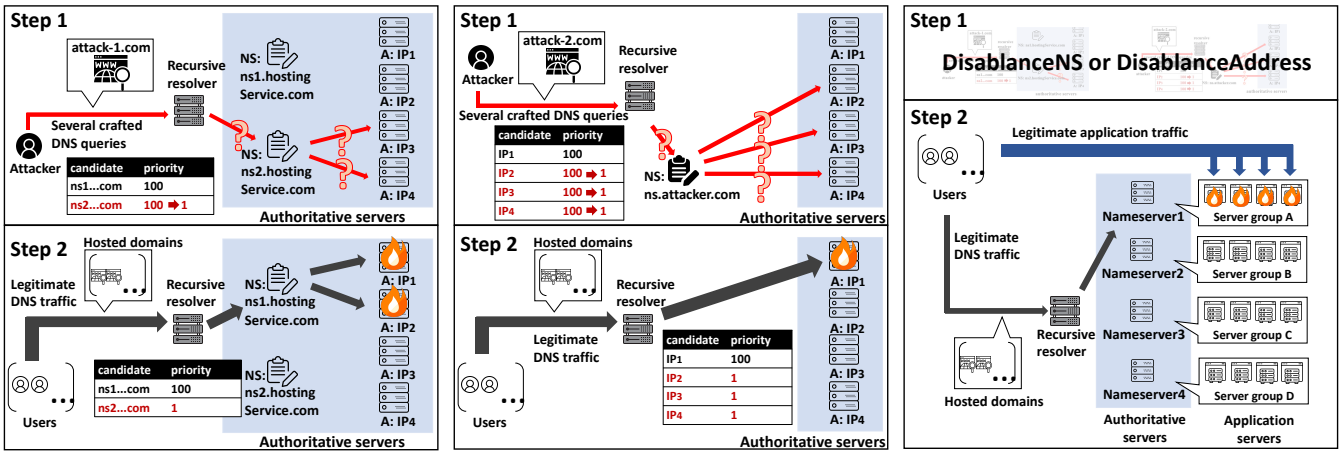
The steps of DisablanceAddress are shown in Figure 4(b). First, adversaries control a domain (e.g., attack-2.com) and configure the A records of their nameservers to point to $IP_{2\text{-}4}$ (attack-2.com in Figure 3). Similar to DisablanceNS, they issue several queries for attack-2.com to the recursive resolver. The resolver then forwards the queries to the authoritative servers at the specified IP address, in which the queries will be ignored since attack-2.com is not hosted on them. The resolver lowers the priorities of $IP_{2\text{-}4}$ and avoids selecting them, causing IP_1 to be overwhelmed handling legitimate queries. Also, benign traffic for all domains hosted on targeted authoritative servers is affected.

Disablance and DNS-based load balancing infrastructure. Disablance can also be exploited to disrupt specific implementations of DNS-based load balancing systems. In such systems, authoritative servers are configured to respond with a different (or partially different) set of resource records pointing to distinctive application server instances. Considering resolvers distribute DNS queries among authoritative servers, the application-layer traffic will also be routed to different instances. However, when authoritative servers are vulnerable to Disablance, clients can only query the nameserver specified by the attacker, receiving the same DNS answer, including the application servers returned by the targeted nameserver. Therefore, legitimate traffic can overload certain instances of application servers.

An example is illustrated in Figure 4(c): a hosted domain is assigned to 4 nameservers ($nameserver_{1\text{-}4}$). Each nameserver is configured to respond differently with a set of records pointing to the application servers (server group A-D). When the recursive resolver is exploited by Disablance, users only receive answers from $nameserver_1$, which will further cause the legitimate application traffic to be directed to an instance in group A, instead of distributing to groups A-D. As a result, access to all domains hosted on targeted nameservers could be affected.

4 ANALYZING DNS RECURSIVE SOFTWARE

In this section, we analyze the impact of Disablance on mainstream DNS implementations, which indicates the behavior of real-world recursive resolvers. Our analysis is two-fold. First, we reveal how Disablance affects popular DNS recursive software by reviewing



(a) Variant I DisablanceNS: diverting legitimate traffic to a single NS record.

(b) Variant II DisablanceAddress: diverting legitimate traffic to a single IP address.

(c) Disablance and DNS-based load balancing infrastructure.

Figure 4: Diagrams of Disablance

their source code (Section 4.1). Second, through simulations, we demonstrate the high efficiency of Disablance (Section 4.2).

In our experiment, we selected a set of software with a high market share (e.g., BIND9 shares 60.2% of identified resolvers [46]), which is a common criterion considered in previous studies [39, 46]. To this end, five DNS implementations are chosen in our study, including BIND9 [38], Unbound [67], PowerDNS [68], Knot [24], and Microsoft DNS (MSDNS) [58, 59].

4.1 Source-code Analysis

We aim to identify how different DNS implementations set the priorities for nameservers. Here, we selected recent versions of four popular open-source DNS implementations: BIND9 (9.18.4), Unbound (1.16.1), Knot Resolver (5.5.1), and PowerDNS Recursor (4.7.1). Each recursor was compiled from the source code in a separate sandbox and linked to GDB remote debugger [30]. We identified the mechanism for setting nameserver priorities by dynamically following the recursors' execution.

Our investigation reveals that BIND9 is affected by Disablance. BIND9 selects the nameserver with the lowest statistical latency when resolving a domain. To achieve this, it maintains smoothed round trip time (SRTT) for every authoritative server. After each query, SRTT is updated according to the latency of this query or set to a random number with a wide range of values if the query is timed out. Since BIND9 shares the SRTT information across all authoritative servers, adversaries can lower the priority of all candidates except for the targeted one by sending unresponsive queries to their domain (Section 3). BIND9 is affected by both DisablanceNS and DisablanceAddress because it records the statistical latency at the IP address level.

PowerDNS is also affected by Disablance. Similar to BIND9, PowerDNS chooses the nameserver with the lowest statistical latency and lowers the priority of those who failed to respond. However, it compares the SRTT of nameservers based on the NS record level instead of the IP address level. As a result, PowerDNS is only affected by the DisablanceNS variant of the attack.

Unbound and Knot Resolver are not affected by the Disablance. Both of them penalize the nameserver that failed to respond. Unbound maintains statistical latency for every delegation. In contrast, Knot Resolver shares statistical information as BIND9 and PowerDNS. However, it tries other candidates with a certain probability and restores its priority immediately when it responds successfully. The algorithm is named ϵ-Greedy (also confirmed in comments in source code). Initially, the algorithm was designed for reinforcement learning and was proposed in 1989 [89]. Therefore, it is unlikely designed to intentionally mitigate the impact of Disablance.

Summary. We found that BIND9 and PowerDNS Recursor are affected by the Disablance. The effectiveness of the attack on these software are affected by the following conditions:

(1) *Number of attacker's queries:* A resolver penalizes the nameservers specified by the adversary when it processes the attacker's DNS query. Hence, the more attacking queries the adversary sends to the resolvers, the more priority the resolver selects a candidate from *unresponsiveNS* decreases. Consequently, the resolver becomes more likely to choose the *targetedNS* for requests.

(2) *Nameserver latency:* BIND9 and PowerDNS use statistical latency as a condition for nameserver selection. They penalize an unresponsive nameserver by assigning an elevated latency value for it. The attack becomes less effective when the actual latency of the nameserver is high. In Section 5, we will detail our measurement of the latency of authoritative servers and confirm that Disablance is powerful in the real world.

(3) *Distribution of legitimate queries that the resolver sends to nameservers:* Two implementations restore the priority of nameservers periodically, allowing the resolver to distribute the queries among all available nameservers. This mechanism coincidentally prevents Disablance from permanently affecting the priority of the nameservers at the resolver. As a result, the more dense distribution (higher frequency) of legitimate queries that the resolver sends to nameservers makes Disablance more impactful. Interestingly, we further confirmed that thousands of legitimate queries are affected when the adversary only sends several packets (Section 4.2).

(4) *Variant:* BIND9 records the status of nameservers at the IP address level. DisablanceNS targets an NS record associated with a set

of IP addresses; therefore, it affects their status simultaneously. DisablanceAddress only targets a single IP address and affects its status. Theoretically, DisablanceNS is more efficient than DisablanceAddress. However, we found no significant difference between the two variants in real-world simulation (Section 4.2).

4.2 Real-world Simulation

We investigate the effectiveness of the attack on popular implementations by conducting two experiments that mimic the real-world environment. For open-source implementations (BIND9 and PowerDNS), we performed a white-box evaluation, allowing us to test different combinations of conditions. In contrast, we ran a black-box experiment on Microsoft DNS, which is closed-source.

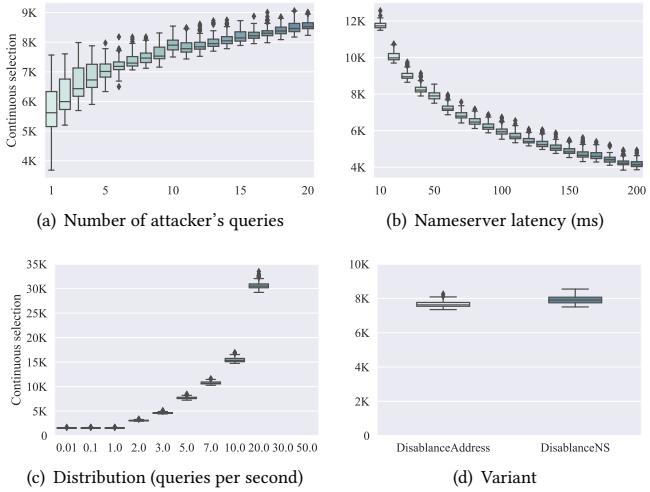
To confirm the effectiveness of the Disablance while both legitimate and malicious queries are sent to a recursor, we evaluated the *continuous selection* of a nameserver after the recursor handles the attacking packets. Specifically, we built a simulative network environment with four vulnerable nameservers and configured a set of hosted domains that point to the nameservers (as shown in Figure 2). Also, two attacking domains for DisablanceNS and DisablanceAddress are configured respectively (as shown in Figure 3). After issuing attacking queries to the recursor, we send a set of queries to mimic the behavior of benign users. Then, the count of the legitimate queries that are continuously sent to the victim nameserver without referring to any other candidates is reported. This metric can measure the *continuous effect* of Disablance under the impact of legitimate traffic.

To examine the effect of each of the four conditions mentioned in 4.1, we first set a group of default values to each one. Then, we changed every condition and observed its impact on the efficiency of Disablance (i.e., continuous selection) while keeping other conditions constant. We set the default values for each condition: 10 attacking packets, 50ms latency of nameservers, uniform distribution of legitimate queries (5 queries per second), and performing DisablanceNS. While some may argue that those values are not strictly representative of real-world traffic, we believe our test covers sufficient situations (mimicked more than 5×10^8 queries to open-source recursors) to evaluate the impact of Disablance.

BIND9 and PowerDNS Recursor. Evaluating the efficiency of Disablance on BIND9 is not straightforward since its algorithm utilizes a random number with a wide range of values (Section 4.1), and repeated experiments are necessary to evaluate the impact of the attack accurately. However, testing the actual resolver repeatedly across different combinations of conditions can be time-consuming. To evaluate efficiently, we extracted the essential code of BIND9 for nameserver selection and executed it in a simulative environment. Following the analysis stated in Section 4.1, we extracted all related code to a `C` program (also utilized by BIND9) without unnecessary modification. The program takes conditions as input and outputs the selection decision. As a result, we can test a combination of conditions by adjusting input and executing the program.

For each combination of conditions, we repeated the experiment 100 times, and the metric of continuous selection was reported. As shown in Figure 5, each condition is discussed below:

(1) Number of attacker's queries: We observed that the attack is more effective and stable when the adversary issues more queries.



(a) Number of attacker's queries (b) Nameserver latency (ms)

(c) Distribution (queries per second) (d) Variant

Figure 5: Evaluation of attacking efficiency to BIND9

However, while receiving only one attacking query, BIND9 continuously sent 5,730 legitimate queries to the targeted nameserver on average without referring any other candidates (Figure 5(a)). The result reveals the lower limit of Disablance is still powerful even when the targeted resolver handles much more benign traffic than malicious traffic.

(2) Nameserver latency: High latency of the targeted nameserver reduces the effect of Disablance. In Section 5.2, we will show that the latency of real-world vulnerable nameserver is less than 50ms in 96.76% of top FQDNs and 68.57% of top SLDs. In this situation, adversaries can cause the resolver to divert more than 7,933 queries on average by only sending ten attacking packets while the latency is less than 50ms (Figure 5(b)).

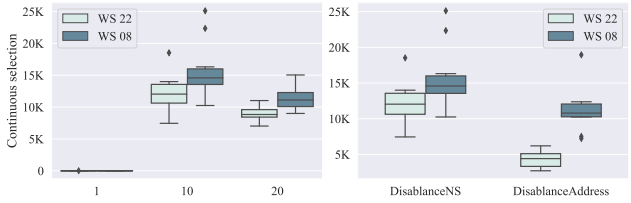
(3) Distribution of legitimate queries that the resolver sends to nameservers: A more dense distribution of legitimate queries leads to a more successful attack. Interestingly, when the frequency is extremely high (e.g., 30 or 50 queries per second), the continuous selection exceeds 50K, which reaches the limit of our simulation (Figure 5(c)). The distribution is determined by the number of hosted domains on the targeted nameserver. In Section 5.2, we demonstrate that many vulnerable top SLDs, along with their hundred million FQDNs, lead the resolver to send queries to nameservers frequently.

(4) Variant: In the simulation, the effects of the two variants have no significant difference (Figure 5(d)).

With the same method, we further evaluated how these conditions affect PowerDNS Recursor. PowerDNS is susceptible to only DisablanceNS variant of the attack (Section 4.1), while other conditions impact PowerDNS in the same way. i.e., more attacking queries, lower nameserver latency, and more dense distribution of legitimate queries result in high efficiency of Disablance.

Microsoft DNS. We evaluated two versions of MSDNS (i.e., Windows Server 2022 (10.0.20348.169) and 2008R2 (6.1.7600). Except for the recent version, we also include the older version since a previous study reported its high market share [46].

We evaluated how Disablance impacts MSDNS through a black-box experiment. MSDNS is a component of an operating system (Windows Server), which is closed-source. Investigating MSDNS using the same experiment on BIND9 and PowerDNS is impractical



(a) Number of attacker's queries (b) Variant

Figure 6: Evaluation of attacking efficiency to MSDNS

and time-consuming. Specifically, BIND9 and PowerDNS are open-source software, and code extraction enabled us to evaluate various sets of conditions. However, to test MSDNS, we need to run the whole operating system in our simulated network environment and reset the DNS cache before every test. Therefore, we simplified the combination of conditions for MSDNS by evaluating two groups of conditions and repeating every test 10 times.

Our results (Figure 6) show that both versions of MSDNS are affected by DisablanceNS and DisablanceAddress. Sending only one attacking query was insufficient in MSDNS, while sending one attack query caused BIND9 to divert thousands of queries to the targeted nameserver. However, after receiving ten attacking queries, MSDNS diverted more than 10K legitimate queries in most cases, while BIND9 only redirected 8K queries on average.

Summary. We conclude the result of the evaluation in Table 1. Three of the five software we analyzed are affected by Disablance. Two of them are affected by both DisablanceNS and DisablanceAddress (i.e., BIND9, MSDNS), while one of them is only affected by DisablanceNS (i.e., PowerDNS). For BIND9 and MSDNS, only several packets are required for an adversary to make thousands of packets diverting to the target nameserver.

Table 1: Summary of analyzing DNS recursive software

Software	Sensitive Variant	Market Share [46]
BIND9	DisablanceNS/Address	60.2+%
Unbound	-	4.8+%
PowerDNS Recursor	DisablanceNS	3.2+%
Microsoft DNS	DisablanceNS/Address	2.5+%
Knot Resolver	-	(no mention)

5 MEASURING EXPLOITABLE TARGETS

In this section, we evaluate the pervasiveness of authoritative servers, resolvers, and application servers that can be impacted by Disablance. To this end, we designed DMiner, which is a semi-automatic framework that finds affected services, and it further estimates the effect that an adversary can cause. We first introduce DMiner (Section 5.1), and then we present the results of our measurement on authoritative servers, resolvers, and application servers (Sections 5.2, 5.3, and 5.4).

5.1 DMiner

DMiner is a framework that identifies vulnerable services. It consists of three components (Figure 7): identifying exploitable authoritative servers along with their domains and service providers, identifying exploitable recursive resolvers, and identifying application servers based on vulnerable DNS infrastructure.

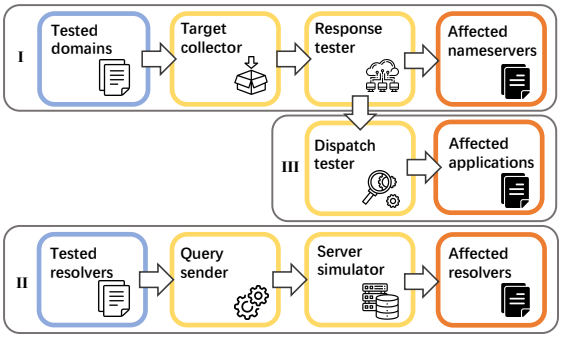


Figure 7: Workflow of DMiner

Module I: Identifying exploitable authoritative servers along with their domains and service providers. This module starts with a list of domains that we are interested in to identify domains and nameservers affected by Disablance. It further discovers the corresponding providers and evaluates the *Amplification Factor (AF)* of the attack. The AF refers to the number of times the amount of legitimate DNS traffic is amplified and diverted to the targeted authoritative server due to Disablance. At a high level, this module first collects all the nameservers for domains in our study. Then checks whether these nameservers can be exploited by Disablance.

The module evaluates the impact of Disablance on both FQDNs and SLDs. This design provides a comprehensive evaluation from different levels, including FQDNs that do not share the same nameservers as their SLDs. To obtain representative targets, we use the top 1 million SLDs from Tranco [48], which have been widely used in previous studies [33]. However, Tranco does not provide information on top-level FQDNs. Therefore, we select the top 1 million FQDNs from SecRank [93].

DMiner requests the `NS` records at the parent zone for each domain, along with the IP addresses of each record. A domain is considered vulnerable only when *all* of its candidate nameservers are vulnerable. As mentioned in Section 3, the attack works against nameservers that are configured to ignore queries for domains that they do not host. However, there could be different reasons behind an unresponsive nameserver (e.g., under maintenance). Hence, we verify two conditions before labeling a nameserver as vulnerable. First, the nameserver ignores queries for a domain that is not hosted on it. Second, it provides responses with correct DNS records to queries for domains it is hosting. For accurate measurement [88], DMiner tests each nameserver from three vantage points at three geographical locations (United States, Germany, and Singapore) and repeats each query five times. DMiner marks a nameserver as vulnerable when it ignores all of the 15 queries that check the first condition and responds correctly to all of the 15 queries for the second condition. DMiner outputs the affected nameservers and the domains hosted on them.

Furthermore, we identify the provider of vulnerable nameservers and the application of affected domains. To identify the provider, DMiner first extracts their apex domain and sorts them by the number of hosted domains. Then, we manually determine the top 10 largest hosting providers through WHOIS data [37], DNS records (`SOA` and `PTR` [62]), databases for nameservers [65], and Google search. Identifying the affected application requires more steps. First, we use the above methods to confirm the provider of affected

domains. Then, we resolve the domain and obtain the IP addresses of application servers. Next, we obtain server banners, opening ports, and community comments from resource search engines [79, 87]. Combining the above information, we can infer the application of the affected domains.

Finally, the amplification factor (AF) of a vulnerable domain will be evaluated. The AF of DisablanceNS is the number of configured NS records for an affected domain, and the AF of DisablanceAddress is the number of IP addresses configured for the nameservers. The results of this module are analyzed in Section 5.2.

Module II: Identifying exploitable recursive resolvers. The module takes a list of resolvers as input and contains a group of machines that simulate exploitable nameservers. It basically sends queries for domains under our control to the targeted resolvers.

DMiner evaluates open resolvers and public recursive services since both of them can be leveraged by attackers. While public recursive services are *willingly* providing recursive services to everyone on the internet (e.g., Google DNS) [76], the open resolvers are *actually* providing services to everyone, no matter whether it meets the expectations of their operators (e.g., misconfigured ISP's resolver) [46, 74]. With ethical considerations, DMiner first filtered out unstable open resolvers from our target list. For the past six months, we scanned active open resolvers in the IPv4 space every two weeks and only kept querying the ones that have been continuously active. At most, we only sent a request to an address every two weeks. Additionally, in order to not disturb any actual nameservers, we equipped DMiner with a set of vulnerable nameservers to simulate real-world ones. DMiner contains authoritative servers with the same hardware capacity and in the same /24, to minimize the difference in their latency. We registered two domains: one acts as a hosted domain and the other as an attacking domain. The queries are sent to subdomains of the hosted domain with different nonce values to simulate that the nameserver is hosting several domains.

To investigate how resolvers react to such vulnerable nameservers, we designed an attack scenario for each resolver of three stages: *waking*, *attacking*, and *verifying*. At the waking stage, DMiner sends three queries for the hosted domains to ensure the resolver is working properly. At the attacking stage, DMiner sends queries for the attacker's domain and attempts to deceive the resolver into selecting only one nameserver in the future. At the verifying stage, DMiner sends queries for hosted domains, and the controlled nameservers report the requests from the resolver. Hence, DMiner can identify whether the attack is successful against the targeted resolver. Based on the difference in the structure between open resolvers and public DNS recursive services, the measurement is adjusted for each type to achieve more accurate results:

(1) Open resolvers. Following the above ethical considerations, DMiner was fed with the IPv4 address of 37,843 resolvers that have been operating stably for half a year. For each open resolver, DMiner is configured to send 10 attacking and 20 verifying queries to check the successfulness of the attack, and at the same time, avoids overloading the resolver. DMiner performs waking, attacking, verifying stages in sequence and labels an open resolver as affected when all of the 20 queries are sent to the targeted nameserver. The maximum of 20 queries is chosen based on the finding of previous studies that showed that DNS software selects the candidate randomly[3, 64, 95] when the latency of all the nameservers candidates are equal, which is the case in our simulated environment. Therefore, theoretically, the probability that a resolver will select the same candidate for successive 20 times is very low (DisablanceNS: $(\frac{1}{2})^{20} \approx 9.5 \times 10^{-7}$, DisablanceAddress: $(\frac{1}{4})^{20} \approx 9.1 \times 10^{-13}$).

(2) Public DNS recursive services. This measurement tests 14 well-known public DNS recursive services that operate 100 IP addresses. For public DNS services, adjustments are applied because it is common to deploy anycast [60, 76, 90, 91] in which multiple nodes in a network share the same address to distribute traffic. Therefore, more attacking packets are required to lower the priority in enough nodes. To achieve a successful attack, two different scenarios of the attack can be executed, which DMiner facilitates to confirm whether a public resolver is affected: *Sequence,* similar to the method for open resolvers, DMiner performs waking, attacking, and verifying stages in sequence. However, to affect as many nodes as possible, it sends 100 attacking packets instead of 10. At the verifying stage, it sends 1000 packets instead of 20 and reports the ratio of packets that are diverted to the targeted nameserver. The other scenario is *Maintain*, an adversary tries to maintain the effect of Disablance by sending attacking packets continuously. After sending 50 attacking packets in waking stage, DMiner performs the attacking and verifying stages parallelly. During 500 seconds, DMiner sends one attacking packet every 10 seconds (50 in total) and sends two testing packets every second (1000 in total). Then, it counts the packets that are received by the targeted nameserver.

DMiner further filters out false-positive cases. Such cases include resolvers that are coincidentally configured to select the nameserver targeted by us. Thus, DMiner performs a comparison check on both measurements without the attacking stage. Precisely, the comparison check follows the same steps used to identify if a target is "exploitable" but without the attacking stage. If the resolver always picks the targeted nameserver, then it is not considered vulnerable.

DMiner utilizes GeoIP database [57] to determine ASes and locations of affected open resolvers. This information can be further used to estimate the amount of traffic an adversary can control. The results are presented in Section 5.3.

Module III: Identifying application servers based on vulnerable DNS infrastructure. This module checks whether Disablance can escalate its impact on the affected domains by exploiting their DNS-based load balancing system (see Section 3.2). Taking the affected domains and authoritative servers as input, DMiner checks if the attack disrupts their load balancing systems. It then evaluates the Amplification Factor (AF), which refers to the number of times the amount of application traffic is diverted to the targeted application server due to Disablance.

For each affected domain, DMiner tests if its nameservers return distinct record sets. Specifically, DMiner requests A records of the domain from each authoritative server repeatedly. Hence, it enumerates all the IP addresses of the application servers that can be returned by each authoritative server of a targeted domain. To avoid enumerating incomplete results or overloading the nameservers, DMiner limits the test rate and extends the test time. On average, DMiner tests 3.85 domains every second. It also shuffles the testing sequence to avoid querying the same server continuously. Finally, the AF of an affected application is reported. Based on the traffic

that attackers can centralize, we present the calculation of AF: let N be the set of all nameservers that the domain has, and A_n ($n \in N$) is the set of application servers returned by a nameserver n, then, the AF is $\frac{|\bigcup_{n\in N} A_n|}{\min_{n\in N} |A_n|}$. The results are presented in Section 5.4.

5.2 Result Analysis of Authoritative Server

In this section, we report our results on authoritative servers. We started our measurement on May 12, 2022. Over the course of one week, DMiner examined the top 1M SecRank FQDNs and the top 1M Tranco SLDs by utilizing three vantage points (located in the United States, Germany, and Singapore respectively). Our results include four aspects: affected domains, authoritative servers, hosting providers, and amplification factors.

Domains. We found that even top domains are at risk from Disablance. In total, 222,370 (22.24%) top FQDNs and 39,392 (3.94%) top SLDs are affected. The distribution of such domains is shown in Table 2. Zooming in the top 100 domains, the proportion of vulnerable FQDNs and SLDs increase to 29% and 11%, respectively. Such vulnerable domains are related to popular applications. The most popular application is the API for a mobile operating system, whose FQDNs are at rank 2 and 9. Short-form video applications, which are related to 26 domains among the top 100 FQDNs, are also heavily affected. Besides, several popular FQDNs that serve E-commerce services (rank 50 and 54, 180, 181, 186, and 200), cloud services (rank 169), and medical services (rank 190 and 191) that require high availability can be impacted by Disablance. Web services on popular SLDs are also vulnerable, including video-sharing services (rank 19) and blogs (rank 26 and 79).

We also analyzed the differences and intersections in the results obtained from the two domain lists. First, we examined the vulnerable SecRank FQDNs and identified 91,903 SLDs that are shared by 27.97% of the total 328,528 FQDNs. This proportion was higher than that of vulnerable FQDNs and SLDs. Then, we focused on the SLDs obtained from affected SecRank FQDNs and found that 22.26% (8,770) of the vulnerable Tranco SLDs overlapped. Interestingly, an SLD serving an E-commerce site was represented in both domain lists and had the most FQDNs detected (9,020 SecRank FQDNs belonging to the Tranco SLD were found).

We further estimated that DNS traffic could be diverted by Disablance. To do this, we collaborated with one of the largest DNS providers, 114DNS [1, 14], and collected passive DNS data from its public recursive service. During the same one-week period in which DMiner examined vulnerable domains, we searched for the responses to queries for FQDNs belonging to affected SLDs and observed 61,601,477 unique FQDNs. While these results may not accurately represent the FQDNs affected by Disablance, we believe they demonstrate the massive DNS traffic that an adversary could manipulate.

Authoritative servers. In top 1M FQDNs, 5,623 out of 47,925 (11.73%) authoritative servers were exploitable, while 13,964 out of 317,222 (4.40%) authoritative servers for domains in top 1M SLDs were found vulnerable. We confirmed in Section 4.1 that the nameserver latency is a factor affecting the efficiency of Disablance. Here we demonstrate that Disablance is powerful when considering the factor of latency. Our investigation reveals that 96.76% of top FQDNs and 68.57% of top SLDs have latency that is lower than 50ms. Assuming adversaries are executing the attack through the most popular recursive implementation (BIND9, sharing 60.2% of identified open resolvers in the wild [46]), they can divert more than 7,933 queries on average to the targeted nameserver by sending only 10 attacking packets when the nameservers' latency is lower than 50ms (as shown in Figure 5(b)).

Table 2: Distribution of affected domains

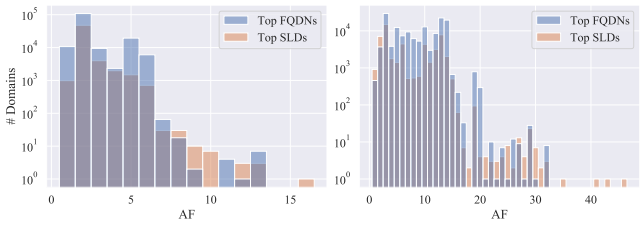
Top	10	100	1K	10K	100K	1M
# FQDN	20%	29%	34.7%	26.9%	25.3%	22.2%
# SLD	10%	11%	6.8%	5.5%	4.6%	3.9%

Table 3: Top 10 affected providers for the top sites

Top 1M FQDNs			Top 1M SLDs		
Provider	**Service[a]**	**# Hosting**	**Provider**	**Service[a]**	**# Hosting**
Tencent Cloud	Cloud	**62,607**	Tencent Cloud	Cloud	**8,119**
WANGSU	Cloud	**34,838**	DNS.COM	Cloud	**4,071**
DNS.COM	Cloud	9,949	WANGSU	Cloud	2,738
GNAME	Domain	7,647	GNAME	Domain	1,645
360	Cloud	2,212	Freenom	Domain	580
SFN	Domain	1,920	Danesconames	Domain	390
Baidu Cloud	Cloud	965	Baidu Cloud	Cloud	337
22.cn	Cloud	843	XZ.com	Domain	250
Na.wang	Cloud	623	22.cn	Cloud	226
CNDNS	Cloud	345	Heteml	Cloud	218
Total		**222,370**	**Total**		**39,392**

[a] Domain services include domain hosting and registration, while cloud services include all domain services and other cloud services, such as VPS and CDN.

DNS service providers. Our results show that Disablance can impact the performance of popular providers. Table 3 summarized the affected providers used by the top FQDNs and SLDs. These providers are well-known and can host a large number of domains. For example, Tencent Cloud (DNSPod) [82], a popular provider affected by Disablance, is hosting a considerable number of popular sites (6.26% of top 1M FQDNs and 0.81% of top 1M SLDs). By performing Disablance, an adversary can divert a massive amount of legitimate traffic toward targeted nameservers that affect many clients. We also found that exploitable nameservers can be deployed on popular cloud providers (e.g., VPS of Amazon [10]), despite their domain hosting service are not directly affected by Disablance.



(a) DisablanceNS (b) DisablanceAddress

Figure 8: Amplification factors of Disablance

Amplification Factor. DMiner evaluates AFs for the two types of Disablance (DisablanceNS and DisablanceAddress). The result is shown in Figure 8. The average AF for DisablanceNS is 2.55× for the vulnerable top FQDNs, while it is 2.26× for the vulnerable top SLDs (Up to 13×, 16× for FQDNs and SLDs, respectively). When performing DisablanceAddress, the average of the AF reaches 8.51× for the vulnerable top FQDNs, while it is 6.84× for the vulnerable

top SLDs (up to 32×, 46× for FQDNs and SLDs, respectively). Also, note that the effect of Disablance involves all domains hosted on the targeted nameserver rather than affects a single domain.

We conducted a simulation to confirm that Disablance is capable of amplifying DNS traffic to a theoretical amplification factor (AF) in a real-world environment. We first set up a vulnerable domain with two NSes severing to eight sites with the same configuration as one of the largest vulnerable DNS providers. Then, we deployed machines acting as recursive resolvers, which ran BIND9, PowerDNS, and MSDNS respectively. After sending attacking queries, we calculated the legitimate queries that were continuously sent to the targeted nameserver (the same metric stated in Section 4). We found the actual AF was equal to the theoretical one in all cases we tested.

Case study. We find that domains requiring high availability are suffering a greater amplification impact since they are assigned more nameservers for load balancing. For example, the AF reaches 32× in eight FQDNs serving a financial institution. Similarly, the AF reaches 46× for a vulnerable SLD owned by a technology company.

5.3 Result Analysis of Recursive Resolver

This section analyzes the measurement results of 37,843 stable open resolvers and 14 public DNS services.

Open resolvers. The measurement started on December 14, 2021. During a 10-day period, DMiner examined targeted open resolvers through a vantage point in the United States. Among 37,843 stable IPv4 open resolvers we tested, 14,372 (37.88%) are at risk of Disablance. Specifically, 12,636 (33.39%) and 9,329 (24.65%) resolvers are impacted by DisablanceNS and DisablanceAddress, respectively.

We also found that affected open resolvers are located around the world (as shown in Figure 9). In total, the open resolvers that can be affected by Disablance are distributed in 130 countries, 2,821 cities, and 1,778 ASes, indicating that they are serving a considerable number of users whose DNS traffic can be diverted by Disablance.

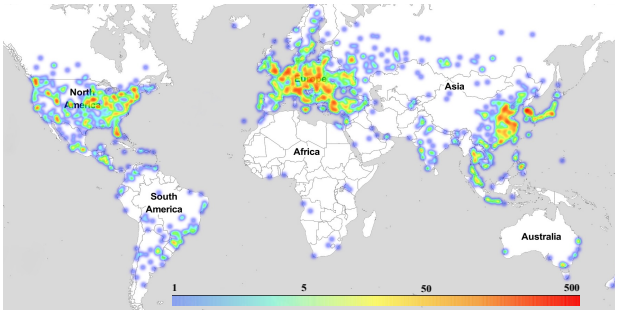


Figure 9: Heat Map of affected open DNS resolvers

Public DNS recursive services. The measurement started on December 29, 2021. During a 70-hour period, DMiner inspected the IP addresses operating by public DNS recursive services through the same vantage point in the US. The result is shown in Table 4. Our result indicates that 45 of 100 of the resolvers operated by 10 of the 14 providers are exploitable including Cloudflare [22], OneDNS [83], and Quad9 [75]. We also found that 10 providers are affected by DisablanceNS in which 8 of them can be impacted with a high success rate. In addition, 8 providers are at risk of DisablanceAddress, of which 4 of them have a high success rate.

We further observed that some providers have only a subset of their servers that are affected. For example, the nodes of Cloudflare at the IPv6 are affected, while those running under IPv4 are not. We discussed with vendors and confirmed the reason behind this: their nodes for different IP addresses are deployed with different software implementations.

Table 4: Statistic of affected IP addresses provided by public DNS recursive services

Vendor	Affected	N_1[a]	N_2[b]	A_1[c]	A_2	#A / #T[d]
Google DNS [32]	NO					0/4
CloudFlare [22]	YES	✔				4/14
OpenDNS [20]	NO					0/12
OneDNS [83]	YES	✔		✔		4/6
Quad9 [75]	YES	✔	✔	✔		11/14
DNS.WATCH [27]	NO					0/4
FreeDNS [29]	YES	✔	✔	✔	✔	5/5
TWNIC Quad 101 [84]	YES	✔	✔	✔	✔	4/4
CleanBrowsing [21]	YES	✔	✔			6/12
Baidu DNS [15]	YES	✔	✔	✔	✔	1/1
UncensoredDNS [85]	YES	✔	✔	✔		1/4
AliDNS [5]	NO					0/4
Alternate DNS [8]	YES	✔	✔	✔		2/4
OpenNIC [69]	YES	✔	✔	✔	✔	7/12
Total						45/100

[a] The existence of targets that are affected by DisablanceNS with a success rate between 75% and 90%.
[b] The success rate is between 90% and 100%.
[c] Targets affected by DisablanceAddress.
[d] The number of affected and total IP addresses.

5.4 Result Analysis of Application Servers with Exploitable Load Balancing Systems

This section reports our findings on vulnerable application servers where the attack can be escalated by exploiting their DNS-based load balancing system.

We used DMiner to identify such cases and calculate the AF (Section 5.1). The three-day measurement was started on May 22, 2022, with the same vantage points utilized in Section 5.2. We found that Disablance can further disrupt the load balancing of 1,010 top FQDNs and 183 top SLDs. In vulnerable FQDNs, we found APIs for popular short-form video applications (rank 48), mobile operating systems (rank 162), E-commerce services (rank 3,322) and IoT management (rank 4,683). Top SLDs can also be affected, including applications for logistics (rank 2,401), banking (rank 4,132), and television services (rank 5,174). DMiner also evaluated AFs while Disablance disrupts the DNS-based load balancing for application servers. The average of the AF is 1.71× for top FQDNs, while it is 2.23× for those in top SLDs (Up to 8×, 4× for top FQDNs and SLDs, respectively).

Case study. The FQDN that suffered the highest AF serves an API service for a news application. The domain name equipped with 24 unique IP addresses for its application servers. Unfortunately, eight authoritative servers exclusively provide a subset of these IP

addresses (three application servers) and are sensitive to Disablance. Consequently, an attacker can divert 8× legitimate upper-layer traffic from users to application servers.

6 MITIGATION

In this section, we propose mitigation solutions for authoritative servers and resolver operators to protect against Disablance.

Authoritative servers. We discussed this issue with the affected hosting services. They confirmed that dropping DNS queries for non-authoritative domains is adopted as a defense mechanism to protect against DNS amplification attacks. Specifically, attackers try to overload ISPs' recursive resolvers. Through IP spoofing, they trick nameservers of hosting services to respond to the queries from "resolvers". Since attackers do not know the exact hosted domains on the nameservers, ignoring the queries for non-authoritative domains is a simple and effective defense mechanism. However, this strategy facilitates Disablance and also violates the DNS specification [12]: " *Failing to respond at all is always incorrect, regardless of the configuration of the server*". By analyzing the response of nameservers that are not affected by Disablance in Section 5.2, we observed that a negative answer (e.g., `REFUSED` and `SERVFAIL`) is returned when a query for a non-authoritative domain is received.

We suggest an accurate response for non-authoritative domains that follow the DNS standard [47], which is to return `REFUSED` with an extended DNS (EDNS) error code (Code 20: Not Authoritative). In case the resolver does not support EDNS [25], we suggest returning `REFUSED` instead of other misleading errors (e.g., `SERVFAIL` is an error response that indicates server failure). Such a strategy was found in some hosting services (e.g., Godaddy [31]) that are not affected by Disablance. These responses do not cause DDoS attacks since it does not generate more responding packets than what the adversary sent. It is challenging to trace the attacker from the side of authoritative servers since they only observe the legitimate resolvers overload specified nameservers through benign traffic. Hence, fixing the root causes is necessary to mitigate Disablance.

We disclosed the issue and shared our recommendations with several hosting services serving large numbers of domains. Accordingly, at the time of writing this paper, Tencent Cloud fixed this issue. TSSNS acknowledged the issue and is preparing to fix it. Amazon also implemented a monitoring technique to protect nameservers deployed on their cloud service since they cannot control them directly.

Recursive resolvers. Although it is theoretically possible to trace and filter out attackers' domains from the resolver side, we believe adjusting the algorithm is far more efficient for fixing the issue at the level of recursive resolvers since these software are installed on most of the affected resolvers. Through discussion, we confirmed that several vendors develop their public DNS recursive services on top of open-source DNS software.

Two reasons cause a resolver to be impacted by Disablance: Sharing the status of nameservers across all authoritative domains and decreasing the priority of a nameserver when the query is timed-out. In Section 4.1, we found Unbound and Knot Resolver are not affected by Disablance. Unbound maintains the status of nameservers per delegation instead of sharing the global status across all domains. Although Unbound is not affected by Disablance, this could cause a new attack by overloading the cache of a resolver. We leave investigating the impact of this attack for future work.

We recommend addressing this issue by adopting the strategy of Knot Resolver. Like BIND9 and PowerDNS, Knot Resolver shares the status of nameservers across all authoritative domains and penalizes the nameserver that failed to respond. However, instead of always selecting the candidate with the lowest statistic latency, Knot tries other candidates with a predetermined probability. If a candidate failed to respond in the past, Knot restores its status once it responds successfully. We believe that this solution does not require much code modification. and maintains the resources of the resolver.

We have reported to all the vendors of the affected software (BIND, PowerDNS, and Microsoft DNS). All of them acknowledged our findings. Unfortunately, they insist that the vulnerability should be fixed by authoritative servers.

7 DISCUSSION

Ethical Considerations. Adhering to guidelines [42, 71] of ethical principles, we carefully design our measurements and report our findings to relevant vendors. The primary ethical concern of this study is avoiding negative consequences while detecting affected servers in the real world. To this end, we limited the number and rate of packets sent and set up our own domains and nameservers for experiments, as detailed in Section 5.

While evaluating real-world resolvers, we only included stable open DNS resolvers. Previous studies [46, 74] have shown that unstable open resolvers tend to be end-user devices (e.g., home routers) with limited hardware capacities. Due to misconfiguration, such resolvers are exposed to the Internet unintentionally and should be removed from a large-scale measurement study. Also, we limited the scope of our open resolvers scanning. Instead of scanning the entire IPv4 space during each scan, we focused only on open resolvers that were identified as alive in the previous scan.

We have also reported our findings and recommendations to many affected vendors, including all developers of the affected DNS software and most of the hosting services mentioned in this paper.

Evidence of existing attacks in the real world. After disclosure, we had online discussions with leading DNS hosting service providers. Surprisingly, one renowned DNS provider, which provided both hosting service and public recursive services, presented evidence of the existing Disablance in their network: They noticed that their recursive resolver kept selecting one of DNSPod's overseas authoritative servers without bothering to query the other 14 candidates (Figure 1). Such a phenomenon caught their attention because it significantly increased the latency of DNS resolution: about 21× more than the normal situation. They first considered BGP routing failure and spent 81 hours finding out the real cause: the resolution of a "misconfigured" domain. After we introduced Disablance, they also acknowledged that such attacks diverting benign traffic could bypass their several existing defense mechanisms against DoS traffic.

DNS attacks targeting load balancing. A previous study presented several related methodologies to attack DNS load balancing [26], including *IP fragmentation*, *rate-limiting*, and *low-rate*

bursts. However, we believe there are significant differences between the two studies.

The targets that fulfill the requirements for Disablance are more prevalent. Specifically, IP fragmentation requires tricking the targeted nameserver into reducing the Maximum Transmission Unit (MTU) to less than 512 bytes. However, less than 0.7% of nameservers for the top 100K Alexa [9] domains meet this condition [99]. Likewise, triggering the rate-limiting of nameservers requires the attacker to send queries at high speed. Even when an attacker sends 4K queries in a second, less than 10% of nameservers for the top 100K Alexa domains will drop more than 90% of queries [56]. Besides, both IP fragmentation and rate-limiting require IP address spoofing. As of Jan 2023, only 8.9% of IPv4 address blocks allow spoofing [18, 54]. Further, the low-rate attack requires accurate time synchronization across multiple agents (e.g., Botnet). This condition is difficult to satisfy in the real world [100].

Compared with the above methodologies, Disablance is a new attack that has not been revealed before and only requires sending a handful of packets to the victim (Section 4). The results reveal that 22.24% of the top FQDNs and 37.88% of open resolvers investigated can be targeted by Disablance. In summary, we believe Disablance is a novel and feasible attack that demands fewer requirements.

Limitations. While testing real-world open resolvers, we did not apply repeated measurements to avoid overloading. Instead, we performed a conservative detection by sending a limited number of packets from a single vantage point. Such a decision could affect accuracy. However, our measurement still identified a considerable number of vulnerable DNS resolvers and demonstrated that Disablance is a realistic threat.

In evaluating vulnerable domains, we selected top SecRank FQDNs and Tranco SLDs. It is possible for some FQDNs to share the same nameservers with their corresponding SLDs, leading to repeated evaluations. However, this design allowed us to examine the issue from different zoom levels. In addition, using Tranco SLDs is more representative than focusing solely on the SLDs within SecRank FQDNs, as the Tranco sites have been widely used in research.

To estimate the potential traffic that can be manipulated, we searched for the FQDNs belonging to vulnerable SLDs in PDNS data. The resulting number is not an exact count of exploitable FQDNs, as some FQDNs are ephemeral, and there could be a time misalignment between our measurement and PDNS data collection. Also, the 114DNS dataset could miss some FQDNs that only serve limited regions. However, we believe the result is representative and confirm that a large amount of traffic is vulnerable to abuse by attackers.

From the perspective of attackers, a limitation of Disablance is that it can not directly target all nameservers and disrupt the resolution. While the priorities of all nameservers are lower, all vulnerable software we tested will still choose a candidate nameserver randomly instead of canceling the resolution. However, we have shown that Disablance is capable of disrupting load balancing, as detailed in section 4 and 5.

8 RELATED WORKS

Attacking load balancing. A significant effort has been devoted to revealing the threats toward load balancing mechanisms for different applications. Allman et al. [7] presented several configurations that undermine the envisioned robustness of DNS ecosystem. And Hao et al. [35] introduced a redirection hijacking that can disable load balancing for CDN. Another research [81] studied a streaming service, Akamai, and found that the service was highly vulnerable to intentional service degradation. Another work [73] suggested that load balancing algorithms should be carefully chosen in different scenarios. Our research presents a novel attack that sabotages the load balancing mechanisms of authoritative servers.

Overloading DNS servers. DNS servers are attractive targets for attackers, and previous studies have disclosed a body of attacks that aim to overload them. Some of them abused open services (e.g., NTP) and generated IP-spoofed requests to reflect answers to the targeted server [44, 77, 78]. Some amplification attacks leveraged complex DNS responses. e.g., long CNAME chains can be exploited to carry amplification attacks against authoritative servers [17]. Additionally, an attacker can craft `NS` records and overload DNS servers [2, 94]. DNSSEC is also potential for amplification attacks [86]. DNS water torture attack was a kind of DDoS attack targeting authoritative servers [4, 55]. Besides, an attack can render resolvers to consume more CPU resources [50]. Providers of DNS services were also the targets of millions of devices in Mirai Botnet [13]. In contrast, the attack introduced in this paper does not require a complex DNS query crafting, nor does it require the attacker to have a complex or high-performance infrastructure.

Analyzing authoritative server selection. Previous studies [3, 64, 95, 97] examined the strategies of common DNS resolvers for selecting authoritative servers. Two methods were common to reveal the strategies: Designing simulation experiments for DNS software, and inspecting outgoing DNS queries from resolvers in the wild. They concluded that most implementations prefer authoritative servers with the lowest latency, while others choose randomly. However, few of them analyzed source code. In this paper, we analyzed the source code of well-known DNS implementations and further understood how Disablance affects them.

9 CONCLUSION

In this paper, we present a new attack, the Disablance, that disrupts the load balancing for authoritative servers. We discovered a prevalent misconfiguration for nameservers and an implementation decision in mainstream DNS software that an adversary can leverage to divert legitimate DNS traffic to a targeted nameserver. We confirmed that Disablance is realistic, efficient, and prevalent. A large number of top sites can be victims. Even popular cloud services can be affected. Besides, a number of stable open resolvers and several well-known public DNS service providers are also exploitable. Moving forward, we provided suggestions to mitigate the threat of Disablance and responsibly disclosed this issue to vendors and service providers. As of the time of writing this paper, vendors, including Tencent Cloud and Amazon, have taken action to fix it.

ACKNOWLEDGMENTS

We thank all anonymous reviewers for their valuable and constructive feedback. This research is supported in part by the National Natural Science Foundation of China (62102218 and 62272265), Alibaba Innovative Research Program (AIR), CCF-Tencent Rhino-Bird Young Faculty Open Research Fund (CCF-Tencent RAGR20230116), and Tsinghua University-China Telecom Corp., Ltd. Joint Research Center for Next Generation Internet Technology Research Fund.

REFERENCES

[1] 114DNS. 2023. 114DNS. http://www.114dns.com/about.html.
[2] Yehuda Afek, Anat Bremler-Barr, and Lior Shafir. 2020. NXNSAttack: Recursive DNS Inefficiencies and Vulnerabilities. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Vancouver, BC, 631–648.
[3] Bernhard Ager, Wolfgang Mühlbauer, Georgios Smaragdakis, and Steve Uhlig. 2010. Comparing DNS Resolvers in the Wild. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement* (Melbourne, Australia) *(IMC '10)*. Association for Computing Machinery, New York, NY, USA, 15–21. https://doi.org/10.1145/1879141.1879144
[4] Akamai. 2019. Whitepaper: DNS Reflection, Amplification, and DNS Water-torture.
[5] Alibaba. 2023. AliDNS. https://www.alidns.com/.
[6] Alibaba. 2023. Domain Name Service. https://www.alibabacloud.com/domain.
[7] Mark Allman. 2018. Comments On DNS Robustness. In *Proceedings of the Applied Networking Research Workshop* (Montreal, QC, Canada) *(ANRW '18)*. Association for Computing Machinery, New York, NY, USA, 16. https://doi.org/10.1145/3232755.3232773
[8] Alternate DNS. 2023. Alternate DNS. https://alternate-dns.com/.
[9] Amazon. 2023. Alexa Top sites. https://www.alexa.com/topsites.
[10] Amazon. 2023. Amazon Web Services (AWS). https://aws.amazon.com.
[11] Mark P. Andrews. 1998. Negative Caching of DNS Queries (DNS NCACHE). RFC 2308. https://doi.org/10.17487/RFC2308
[12] Mark P. Andrews and Ray Bellis. 2020. A Common Operational Problem in DNS Servers: Failure to Communicate. RFC 8906. https://doi.org/10.17487/RFC8906
[13] 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 *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1093–1110. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis
[14] APNIC. 2023. Use of DNS Resolvers for World. https://stats.labs.apnic.net/rvrs/XA?hc=XA&hl=1&hs=0&ht=30&w=1&t=37.
[15] Baidu. 2023. BaiduDNS. https://dudns.baidu.com/index.html.
[16] Sam Burnett and Nick Feamster. 2015. Encore: Lightweight Measurement of Web Censorship with Cross-Origin Requests. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 653–667. https://doi.org/10.1145/2785956.2787485
[17] Jonas Bushart and Christian Rossow. 2018. DNS Unchained: Amplified Application-Layer DoS Attacks Against DNS Authoritatives. In *Research in Attacks, Intrusions, and Defenses*, Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis (Eds.). Springer International Publishing, Cham, 139–160.
[18] CAIDA. 2022. Spoofer. https://spoofer.caida.org/summary.php.
[19] Jianjun Chen, Jian Jiang, Haixin Duan, Nicholas Weaver, Tao Wan, and Vern Paxson. 2016. Host of Troubles: Multiple Host Ambiguities in HTTP Implementations. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 1516–1527. https://doi.org/10.1145/2976749.2978394
[20] CISCO. 2023. OpenDNS. https://www.opendns.com/.
[21] CleanBrowsing. 2023. DNS Content Filtering. https://cleanbrowsing.org/.
[22] Cloudflare. 2023. Cloudflare DNS. https://www.cloudflare.com/dns/.
[23] M. Colajanni, P.S. Yu, and V. Cardellini. 1998. Dynamic Load Balancing in Geographically Distributed Heterogeneous Web Servers. In *Proceedings. 18th International Conference on Distributed Computing Systems (Cat. No.98CB36183)*. 295–302. https://doi.org/10.1109/ICDCS.1998.679729
[24] CZ.NIC. 2023. Knot Resolver. https://www.knot-resolver.cz/.
[25] Joao da Silva Damas, Michael Graff, and Paul A. Vixie. 2013. Extension Mechanisms for DNS (EDNS(0)). RFC 6891. https://doi.org/10.17487/RFC6891
[26] Tianxiang Dai, Haya Shulman, and Michael Waidner. 2021. Let's Downgrade Let's Encrypt. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 1421–1440. https://doi.org/10.1145/3460120.3484815
[27] DNS.WATCH. 2023. DNS.WATCH. https://dns.watch/.
[28] Viktor Dukhovni and Wes Hardaker. 2015. The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance. RFC 7671. https://doi.org/10.17487/RFC7671
[29] FreeDNS. 2023. FreeDNS. https://freedns.zone/en/.
[30] GNU. 2023. GDB: The GNU Project Debugger. https://www.gnu.org/software/gdb/.
[31] Godaddy. 2023. Domain Names, Websites, and Hosting. https://sg.godaddy.com/.
[32] Google. 2023. Google Public DNS. https://developers.google.com/speed/public-dns.
[33] Google scholar. 2023. Studies cited Tranco list. https://scholar.google.com/scholar?cites=149969834840507596.
[34] Brij B Gupta and Omkar P Badve. 2017. Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment. *Neural Computing and Applications* 28, 12 (2017), 3655–3682.
[35] Shuai Hao, Yubao Zhang, Haining Wang, and Angelos Stavrou. 2018. End-Users Get Maneuvered: Empirical Analysis of Redirection Hijacking in Content Delivery Networks. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 1129–1145. https://www.usenix.org/conference/usenixsecurity18/presentation/hao
[36] Paul E. Hoffman and Jakob Schlyter. 2012. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698. https://doi.org/10.17487/RFC6698
[37] ICANN. 2023. Registration data lookup tool. https://lookup.icann.org/.
[38] ISC. 2023. BIND9. https://www.isc.org/bind/.
[39] Philipp Jeitner and Haya Shulman. 2021. Injection Attacks Reloaded: Tunnelling Malicious Payloads over DNS. In *30th USENIX Security Symposium (USENIX Security 21)*. 3165–3182.
[40] 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
[41] Eric Dean Katz, Michelle Butler, and Robert McGrath. 1994. A Scalable HTTP Server: The NCSA Prototype. *Computer Networks and ISDN systems* 27, 2 (1994), 155–164.
[42] Erin Kenneally and David Dittrich. 2012. The Menlo Report: Ethical principles guiding information and communication technology research. *Available at SSRN 2445102* (2012).
[43] Scott Kitterman. 2014. Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1. RFC 7208. https://doi.org/10.17487/RFC7208
[44] Lukas Krämer, Johannes Krupp, Daisuke Makita, Tomomi Nishizoe, Takashi Koide, Katsunari Yoshioka, and Christian Rossow. 2015. AmpPot: Monitoring and Defending Against Amplification DDoS Attacks. In *Research in Attacks, Intrusions, and Defenses*, Herbert Bos, Fabian Monrose, and Gregory Blanc (Eds.). Springer International Publishing, Cham, 615–636.
[45] Murray Kucherawy, Dave Crocker, and Tony Hansen. 2011. DomainKeys Identified Mail (DKIM) Signatures. RFC 6376. https://doi.org/10.17487/RFC6376
[46] Marc Kührer, Thomas Hupperich, Jonas Bushart, Christian Rossow, and Thorsten Holz. 2015. Going Wild: Large-Scale Classification of Open DNS Resolvers. In *Proceedings of the 2015 Internet Measurement Conference* (Tokyo, Japan) *(IMC '15)*. Association for Computing Machinery, New York, NY, USA, 355–368. https://doi.org/10.1145/2815675.2815683
[47] Warren "Ace" Kumari, Evan Hunt, Roy Arends, Wes Hardaker, and David C Lawrence. 2020. Extended DNS Errors. RFC 8914. https://doi.org/10.17487/RFC8914
[48] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS 2019)*. https://doi.org/10.14722/ndss.2019.23386
[49] Jun Li, Minho Sung, Jun Xu, and Li Li. 2004. Large-scale IP traceback in high-speed Internet: practical techniques and theoretical foundation. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004.* 115–129. https://doi.org/10.1109/SECPRI.2004.1301319
[50] Xiang Li, Wei Xu, Baojun Liu, Mingming Zhang, Zhou Li, Jia Zhang, Deliang Chang, Xiaofeng Zheng, Chuhan Wang, Jianjun Chen, Haixin Duan, and Qi Li. 2024. TuDoor Attack: Systematically Exploring and Exploiting Logic Vulnerabilities in DNS Response Pre-processing with Malformed Packets. In *Proceedings of 2024 IEEE Symposium on Security and Privacy (Oakland S&P '24)*.
[51] Wilson Lian, Eric Rescorla, Hovav Shacham, and Stefan Savage. 2013. Measuring the Practical Impact of DNSSEC Deployment. In *22nd USENIX Security Symposium (USENIX Security 13)*. USENIX Association, Washington, D.C., 573–588. https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/lian

[52] Baojun Liu, Chaoyi Lu, Haixin Duan, Ying Liu, Zhou Li, Shuang Hao, and Min Yang. 2018. Who is answering my queries: Understanding and characterizing interception of the {DNS} resolution path. In *27th USENIX Security Symposium (USENIX Security 18)*. 1113–1128.

[53] Xin Liu, Ang Li, Xiaowei Yang, and David Wetherall. 2008. Passport: Secure and Adoptable Source Authentication.. In *USENIX Symposium on Networked Systems Design and Implementation*, Vol. 8. 365–378.

[54] Matthew Luckie, Robert Beverly, Ryan Koga, Ken Keys, Joshua A. Kroll, and k claffy. 2019. Network Hygiene, Incentives, and Regulation: Deployment of Source Address Validation in the Internet. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 465–480. https://doi.org/10.1145/3319535.3354232

[55] Xi Luo, Liming Wang, Zhen Xu, Kai Chen, Jing Yang, and Tian Tian. 2018. A Large Scale Analysis of DNS Water Torture Attack. In *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence* (Shenzhen, China) *(CSAI '18)*. Association for Computing Machinery, New York, NY, USA, 168–173. https://doi.org/10.1145/3297156.3297272

[56] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. 2020. DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, USA) *(CCS '20)*. Association for Computing Machinery, New York, NY, USA, 1337–1350. https://doi.org/10.1145/3372297.3417280

[57] MaxMind. 2023. GeoIP Databases. https://www.maxmind.com/en/geoip2-services-and-databases.

[58] Microsoft. 2023. Windows Server 2008 R2. https://microsoft.fandom.com/wiki/Windows_Server_2008_R2.

[59] Microsoft. 2023. Windows Server 2022. https://microsoft.fandom.com/wiki/Windows_Server_2022.

[60] Walter Milliken, Trevor Mendez, and Dr. Craig Partridge. 1993. Host Anycasting Service. RFC 1546. https://doi.org/10.17487/RFC1546

[61] Jelena Mirkovic and Peter Reiher. 2004. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *SIGCOMM Comput. Commun. Rev.* 34, 2 (apr 2004), 39–53. https://doi.org/10.1145/997150.997156

[62] P. Mockapetris. 1987. Domain Names - Concepts and Facilities. RFC 1034. https://doi.org/10.17487/RFC1034

[63] 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. https://doi.org/10.1145/3419394.3423625

[64] Moritz Müller, Giovane C. M. Moura, Ricardo de O. Schmidt, and John Heidemann. 2017. Recursives in the Wild: Engineering Authoritative DNS Servers. In *Proceedings of the 2017 Internet Measurement Conference* (London, United Kingdom) *(IMC '17)*. Association for Computing Machinery, New York, NY, USA, 489–495. https://doi.org/10.1145/3131365.3131366

[65] MYIP.MS. 2023. DNS Nameservers for 6,000,000 Top World Websites. https://myip.ms/browse/dns/Sites_DNS_Nameservers.

[66] Namecheap. 2023. Namecheap Hosting. https://www.namecheap.com/hosting/.

[67] NLnet Labs. 2023. Unbound. https://www.nlnetlabs.nl/projects/unbound/about/.

[68] Open-Xchange. 2023. PowerDNS Recursor. https://www.powerdns.com/recursor.html.

[69] OpenNIC. 2023. OpenNIC Project. https://www.opennic.org/.

[70] Oracle. 2023. DynDNS. https://account.dyn.com/.

[71] Craig Partridge and Mark Allman. 2016. Ethical Considerations in Network Measurement Papers. *Commun. ACM* 59, 10 (sep 2016), 58–64. https://doi.org/10.1145/2896816

[72] Michael A. Patton, Scott O. Bradner, Robert Elz, and Randy Bush. 1997. Selection and Operation of Secondary DNS Servers. RFC 2182. https://doi.org/10.17487/RFC2182

[73] Olga Pearce, Todd Gamblin, Bronis R. de Supinski, Martin Schulz, and Nancy M. Amato. 2012. Quantifying the Effectiveness of Load Balance Algorithms. In *Proceedings of the 26th ACM International Conference on Supercomputing* (San Servolo Island, Venice, Italy) *(ICS '12)*. Association for Computing Machinery, New York, NY, USA, 185–194. https://doi.org/10.1145/2304576.2304601

[74] Paul Pearce, Ben Jones, Frank Li, Roya Ensafi, Nick Feamster, Nick Weaver, and Vern Paxson. 2017. Global Measurement of DNS Manipulation. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 307–323. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/pearce

[75] Quad9. 2023. Service Addresses. https://www.quad9.net/service/service-addresses-and-features.

[76] 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

[77] Christian Rossow. 2014. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society. https://www.ndss-symposium.org/ndss2014/amplification-hell-revisiting-network-protocols-ddos-abuse

[78] Giovanni Schmid. 2021. Thirty Years of DNS Insecurity: Current Issues and Perspectives. *IEEE Communications Surveys Tutorials* 23, 4 (2021), 2429–2459. https://doi.org/10.1109/COMST.2021.3105741

[79] Shodan. 2023. Shodan Search Engine. https://www.shodan.io/.

[80] RIPE NCC Staff. 2015. Ripe Atlas: A Global Internet Measurement Network. *Internet Protocol Journal* 18, 3 (2015).

[81] Ao-Jan Su and Aleksandar Kuzmanovic. 2008. Thinning Akamai. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement* (Vouliagmeni, Greece) *(IMC '08)*. Association for Computing Machinery, New York, NY, USA, 29–42. https://doi.org/10.1145/1452520.1452525

[82] Tencent. 2023. Tencent Cloud. https://intl.cloud.tencent.com/?lang=en.

[83] Threatbook. 2023. OneDNS. https://www.onedns.net/.

[84] TWNIC. 2023. Quad101. https://101.101.101.101/.

[85] UncensoredDNS. 2023. UncensoredDNS. https://blog.uncensoreddns.org/dns-servers/.

[86] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. 2014. DNSSEC and Its Potential for DDoS Attacks: A Comprehensive Measurement Study. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (Vancouver, BC, Canada) *(IMC '14)*. Association for Computing Machinery, New York, NY, USA, 449–460. https://doi.org/10.1145/2663716.2663731

[87] VirusTotal. 2023. VirusTotal. https://www.virustotal.com/.

[88] Gerry Wan, Liz Izhikevich, David Adrian, Katsunari Yoshioka, Ralph Holz, Christian Rossow, and Zakir Durumeric. 2020. On the Origin of Scanning: The Impact of Location on Internet-Wide Scans. In *Proceedings of the ACM Internet Measurement Conference* (Virtual Event, USA) *(IMC '20)*. Association for Computing Machinery, New York, NY, USA, 662–679. https://doi.org/10.1145/3419394.3424214

[89] Christopher John Cornish Hellaby Watkins. 1989. Learning from delayed rewards. (1989).

[90] Wikipedia. 2023. Load balancing (computing). https://en.wikipedia.org/wiki/Load_balancing_(computing).

[91] Wikipedia. 2023. Public Recursive Name Server. https://en.wikipedia.org/wiki/Public_recursive_name_server.

[92] Wikipedia. 2023. Residential Proxy. https://en.wikipedia.org/wiki/Proxy_server.

[93] Qinge Xie, Shujun Tang, Xiaofeng Zheng, Qingran Lin, Baojun Liu, Haixin Duan, and Frank Li. 2023. Building an Open, Robust, and Stable Voting-Based Domain Top List. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 625–642. https://www.usenix.org/conference/usenixsecurity22/presentation/xie

[94] Wei Xu, Xiang Li, Chaoyi Lu, Baojun Liu, Jia Zhang, Jianjun Chen, Tao Wan, and Haixin Duan. 2023. TsuKing: Coordinating DNS Resolvers and Queries into Potent DoS Amplifiers. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*.

[95] Yingdi Yu, Duane Wessels, Matt Larson, and Lixia Zhang. 2012. Authority Server Selection in DNS Caching Resolvers. *ACM SIGCOMM Computer Communication Review* 42, 2 (2012), 80–86.

[96] Saman Taghavi Zargar, James Joshi, and David Tipper. 2013. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys & Tutorials* 15, 4 (2013), 2046–2069. https://doi.org/10.1109/SURV.2013.031413.00127

[97] Fenglu Zhang, Chaoyi Lu, Baojun Liu, Haixin Duan, and Ying Liu. 2022. Measuring the Practical Effect of DNS Root Server Instances: A China-Wide Case Study. In *Passive and Active Measurement*, Oliver Hohlfeld, Giovane Moura, and Cristel Pelsser (Eds.). Springer International Publishing, Cham, 247–263.

[98] Fenglu Zhang, Yunyi Zhang, Baojun Liu, Eihal Alowaisheq, Lingyun Ying, Xiang Li, Zaifeng Zhang, Ying Liu, Haixin Duan, and Min Zhang. 2023. Wolf in Sheep's Clothing: Evaluating the Security Risks of the Undelegated Record on DNS Hosting Services. In *Proceedings of the ACM Internet Measurement Conference (IMC '23)*. Association for Computing Machinery, Montreal, Canada.

[99] Xiaofeng Zheng, Chaoyi Lu, Jian Peng, Qiushi Yang, Dongjie Zhou, Baojun Liu, Keyu Man, Shuang Hao, Haixin Duan, and Zhiyun Qian. 2020. Poison over Troubled Forwarders: A Cache Poisoning Attack Targeting DNS Forwarding Devices. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Vancouver, BC, 577–593.

[100] Wu Zhijun, Li Wenjing, Liu Liang, and Yue Meng. 2020. Low-rate DoS Attacks, Detection, Defense, and Challenges: A Survey. *IEEE access* 8 (2020), 43920–43943.