TsuKing: Coordinating DNS Resolvers and Queries into Potent DoS Amplifiers

Wei Xu[*]
Tsinghua University
Beijing, China
xu-w21@mails.tsinghua.edu.cn

Xiang Li[*]
Tsinghua University
Beijing, China
x-l19@mails.tsinghua.edu.cn

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

Baojun Liu[†]
Tsinghua University
Beijing, China
lbj@tsinghua.edu.cn

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

Jia Zhang
Tsinghua University
Zhongguancun Laboratory
Beijing, China
zhangjia2017@tsinghua.edu.cn

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

Tao Wan
CableLabs
Louisville, United States
wantao@gmail.com

ABSTRACT

In this paper, we present a new DNS amplification attack, named TsuKing. Instead of exploiting individual DNS resolvers independently to achieve an amplification effect, TsuKing deftly coordinates numerous vulnerable DNS resolvers and crafted queries together to form potent DoS amplifiers. We demonstrate that with TsuKing, an initial small amplification factor can increase exponentially through the internal layers of coordinated amplifiers, resulting in an extremely powerful amplification attack. TsuKing has three variants, including DNSRetry, DNSChain, and DNSLoop, all of which exploit a suite of inconsistent DNS implementations to achieve enormous amplification effect. With comprehensive measurements, we found that about 14.5% of 1.3M open DNS resolvers are potentially vulnerable to TsuKing. Real-world controlled evaluations indicated that attackers can achieve a packet amplification factor of at least 3,700× (DNSChain). We have reported vulnerabilities to affected vendors and provided them with mitigation recommendations. We have received positive responses from 6 vendors, including Unbound, MikroTik, and AliDNS, and 3 CVEs were assigned. Some of them are implementing our recommendations.

CCS CONCEPTS

• **Security and privacy** → **Network security**; *Denial-of-service attacks*; • **Networks** → *Naming and addressing*.

[*]Both authors contributed equally to the paper.
[†]Both authors are corresponding authors.

KEYWORDS

Domain name system; DNS security; Denial of service

1 INTRODUCTION

As a fundamental infrastructure of the Internet, Domain Name System (DNS) is responsible for mapping domain names and IP addresses. It provides support for multiple network services like cloud hosting [57], certificate issuance [58] and email authentication [52].

However, the design of DNS lacks adequate security considerations despite its critical importance. One of the underlying problems is that the a response packet's size can be considerably larger than the query's. Moreover, DNS is built over stateless UDP, which attackers may exploit to conduct reflection amplification attacks by spoofing source addresses. As a result, tens of millions of domains become victims of DNS-based Denial of Service (DoS) attacks [9, 48].

With the evolution of the DNS ecosystem, including the development of public DNS services [44], the structure of DNS has become increasingly complex, providing more opportunities for launching DoS attacks via DNS. For example, according to Randall et al. [44], Google Public DNS has numerous egress IPs around the world, each of which could be exploited to amplify the query traffic. Although new technologies such as DNS Cookie [4] have been proposed to mitigate such attacks, DNS continues to confront new security risks of being targeted by or utilized to launch amplification attacks, such as TsuNAME [39] and other amplification attacks [51, 55].

Prior work. Regarding amplification attacks, there are basically two ways to increase the amplification factor: *(i)* one is to increase the size of DNS response packet to amplify bandwidth, like attacks

using ANY, TXT, and DNSSEC queries [10, 40, 51]; *(ii)* the other is to increase the number of outgoing query packets, such as exploiting vulnerabilities of NS and CNAME record processing [7, 39]. However, these works primarily rely on individual vulnerable resolvers to conduct amplification attacks, in addition to the use of botnets and source address spoofing. The security community has designed corresponding techniques to reduce the amplification impact, such as rejecting the ANY query or restricting the amount of queries [7, 14, 19, 39], thus launching those attacks becomes more difficult.

Our study. In this paper, we propose a new DNS amplification attack, named TsuKing. Instead of focusing on a single target with large amplification capability, TsuKing coordinates a large number of vulnerable DNS resolvers that may have small amplification power into potent DoS amplifiers by delicately creating *a new DNS resolution path.* Along this path, the initial small amplification factor grows exponentially to thousand times larger. TsuKing can attack a wide range of victims with low cost. TsuKing utilizes two key techniques to perform the DNS amplification attack, which can be summarized as *Tsu* (Tsunami) and *King* [24]. The Tsu technique increases the amplification effect, while the King technique coordinates DNS resolvers together. Like King, which uses an authoritative name server to direct DNS query to measure latency between any IP addresses, TsuKing uses such a name server to direct malicious DNS queries and responses. The adversary employs carefully crafted NS responses to manipulate DNS queries to be forwarded between different resolvers, producing a chain or loop of queries (*King*). On the other hand, the resolver's retry mechanism along with multiple egresses generates additional packets, resulting in amplification (*Tsu*). This retry process is replicated continuously down the chain (loop), leading to exponentially hierarchical amplification level by level (See Section 3).

Based on these techniques, TsuKing has three attack variants: *(i)* DNSRetry makes resolvers retry aggressively to send a large number of queries to the victim directly. *(ii)* DNSChain coordinates numbers of vulnerable resolvers into a query chain to hierarchically amplify queries. It can launch a DDoS attack with only two devices, making it more cost-effective than traditional botnet-based attacks. *(iii)* DNSLoop assembles numerous vulnerable resolvers into query loops that process the same request unlimitedly (See Section 4).

To investigate the impact of TsuKing in the wild, we collected 1.3M open DNS resolvers by carefully scanning IPv4 address space with ethical considerations. Our analysis indicated that about 14.5% of DNS resolvers are vulnerable to the TsuKing exploit. Specially, we found that some popular public DNS services, such as 114DNS [3], are vulnerable to TsuKing. Through experiment and analysis, we showed that many widely-used DNS implementations pose potential risks to the TsuKing attack (See Section 5).

We conducted controlled experiments in the real-world on each attack variant to evaluate the feasibility and provide in-depth analysis. Results showed that attackers can simply launch three attack variants and TsuKing can achieve a packet amplification factor (PAF) of at least *3,700×* (the DNSChain attack) even in small-scale experiments. Specifically, the three attack variants reached a *638* PAF (DNSRetry), *3,700* PAF (DNSChain), and forwarded *43,190* times for an attack query (DNSLoop, in 24h) (See Section 6).

We have recommended mitigation solutions and provided vulnerability reports to affected vendors and service providers. So
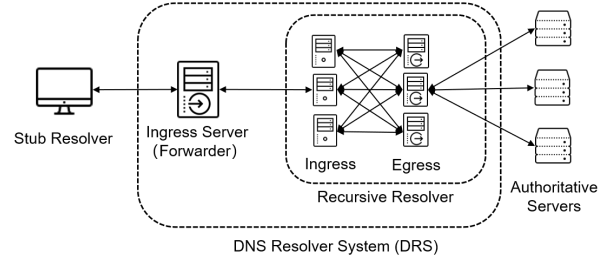

Figure 1: DNS resolver system and resolution paths.

far, we have received responses from 6 vendors, including DNS software and services, such as Unbound, MikroTik, and AliDNS, all of whom have confirmed the vulnerability and committed to providing patch codes to fix it. In addition, 3 new CVE numbers have been assigned to TsuKing (See Section 7 and Section 8).

Contributions. The contributions of this paper are as follows:

- *New threat model.* We proposed a novel threat model for DNS amplification attacks: TsuKing that creates amplification attacks by coordinating DNS resolvers and queries.

- *Measurement and evaluation.* We measured practical factors affecting TsuKing comprehensively and identified a large number of vulnerable DNS resolvers with extensive evaluations.

- *Mitigation and disclosure.* We provided mitigation resolutions and vulnerability reports to all affected vendors, and have received their confirmation and patching plans.

2 DNS RESOLVER STRUCTURE AND AMPLIFICATION ATTACKS

In this section, we first introduce the background of the DNS infrastructure [1, 2]. Then, we review the techniques used in previous DNS amplification attacks that are related to our attack.

2.1 The Structure of DNS Resolver System

Traditionally, the client-side DNS infrastructure is very simple: a *stub resolver* of an end host sends DNS queries to *recursive resolvers*, which in turn query *authoritative servers* and bring back the final answers. However, as the Internet evolves, the client-side infrastructure now includes multiple types and layers of DNS resolvers [46]. The most recognized example is *DNS forwarders*, which forward incoming queries to their designated upstream servers. Forwarders are often deployed between stub and recursive resolvers (e.g., appearing as home routers [59]) or even between forwarders themselves, thus adding layers to the DNS infrastructure. Meanwhile, large public recursive resolvers (e.g., Google Public DNS [23]) have developed from standalone server endpoints into complex systems [44], often comprising load balancing mechanisms (e.g., anycast [22]), clusters of caching servers, and multiple backend resolvers that directly contact authoritative servers.

Figure 1 illustrates a typical DNS resolution path for a client-side DNS query: *(i)* the end user sends a DNS query to an *ingress* DNS server (typically a DNS forwarder); *(ii)* the query passes one or more layers of forwarders, the last of which directs it to the anycast endpoint of a recursive resolver; *(iii)* the DNS query goes through the caching servers, transfers from one or more *egress* servers, and finally arrives at the authoritative servers.

Definition of DNS Resolver System (DRS). From the perspective of an end user, the nearest resolver, DNS forwarder or ingress resolver of a public DNS resolver, appears as a traditional recursive resolver that hides everything else in the infrastructure. In this paper, we refer to an ingress server (e.g., an open DNS server), together with everything between it and authoritative servers (i.e., all upstream servers and egress servers in the resolution paths), as a *DNS resolver system (DRS)*. Under this definition, a DRS includes only one ingress server and one or multiple egress servers. A standalone recursive resolver itself serves as both ingress and egress server, and is considered a minimal DRS.

2.2 DNS Amplification Attacks

As DNS responses are inherently larger than queries, attackers can send carefully-crafted DNS requests that aim to trigger massive amounts of outgoing traffic from DNS servers. The effect is often measured by bandwidth amplification factor (BAF) and packet amplification factor (PAF), defined by Rossow [45]:

$$BAF = \frac{len(payload) \; amplifier \; to \; victim}{len(payload) \; attacker \; to \; amplifier} \quad (1)$$

$$PAF = \frac{number \; of \; packets \; (amplifier \; to \; victim)}{number \; of \; packets \; (attacker \; to \; amplifier)} \quad (2)$$

There are two types of techniques leveraged by previous attacks to increase the amplification factor, as summarized below.
Type I: Increasing the DNS response size. Increasing the size of single DNS responses directly enlarges BAF. One common approach is to exploit query types that yield large responses, e.g., ANY [10, 40] (asking the DNS server to return *all* records for the queried domain) and TXT (text record). While ANY and TXT are still supported by 3~8% of open DNS resolvers [54], the number is expected to decline as mitigation has been proposed, e.g., minimal-sized responses for type-ANY queries [6]. Another approach is to leverage new DNS extensions, which put more records in DNS responses, e.g., EDNS(0) [20] and DNSSEC [51]. Furthermore, by proposing the AmpMap measurement framework for amplification attacks, a recent study [38] finds new query patterns that facilitate large responses (e.g., LOC and URI query types).

As summarized in Table 1, we find that this type of DNS amplification attack can yield hundreds of BAF. However, DDoS attackers typically send DNS queries in parallel from botnets instead of one single host, and need a wide array of DNS servers as amplifiers.
Type II: Increasing the amount of packets. Another approach to increasing PAF is to have the DRS produce more packets. As *iDNS Attack* [34] proposed, the malicious NS (which DNS server is the domain's authoritative server) records can be utilized to cause the resolver to generate additional responses to a query. In this way, *DNS Unchained* [15] and *TsuNAME* [39] are techniques for increasing the number of DRS requests to authoritative servers through the use of NS or CNAME (name alias) records. By configuring records to form dependencies, DRSes will repeatedly launch queries to the authoritative server when attempting to resolve a domain, even becoming stuck in a loop. The other option proposed by *NXNSAttack* [7] is to load the response with a large number of malicious NS records, which the DRS may attempt one by one, causing a huge number of requests to be sent to the victim.

Table 1: Study of DNS amplification attacks.

Attack	Record type	AF[1]
DNS reflection amplification attack [10, 51]	ANY DNSKEY	200+ BAF 50+ BAF
TsuNAME [39]	CNAME or NS	500 PAF
NXNSAttack [7]	NS	≤3,154 PAF
DNS unchained [15]	CNAME or NS	8.51 PAF
Routing loops as mega Amplifiers [41]	*	927,726 PAF[2]
DNSRetry[3]	NS	638+ PAF
DNSChain[3]	NS	≥3,700 PAF
DNSLoop[3]	NS	≥43,190 PAF[4]

[1]: Amplification factor.
[2]: In measurements, only 64 targets could achieve this level of amplification.
[3]: TsuKing attack variants (more detail in Section 4).
[4]: The times of an attack query forwarded in 24h.

There is also another technique that involves leveraging routing loops observed by Yevheniya et al. [41]. When a middle box exists in a network loop, it may process the same DNS request multiple times, leading to a substantial number of responses sent to victims.

As shown in Table 1, under certain conditions, these attacks can achieve amplification of thousand times against the original query.
Our work. Our threat model also uses NS records to conduct attacks. However, instead of increasing the number of retries on a single DRS, we coordinate multiple DRSes to form attacks.

3 THREAT MODEL OF TSUKING

DNS has been one of the most abused protocols for amplification attacks [9], affecting the Internet with an amplification factor from around 10 to over 3,000. The security community has developed specific and delicate methods to lessen the impact [7, 14, 19, 39], such as rejecting the ANY query or restricting the amount of queries (as shown in the column "*BAF related*" and "*PAF related*" of Table 2).

However, due to the diverse DNS implementations and inconsistent behaviors of different DNS servers, it is important to investigate whether it is possible to launch a potent amplification attack utilizing resolvers with small PAFs instead of resolvers with large amplification risks. This motivates our proposed TsuKing model.

Below, we first describe our key observations to support our attacks. Then, we show our threat model and attack overview. Finally, we give practical considerations and attack comparisons.

3.1 Key factors of TsuKing

Resolvers determine whether and how a DNS query should be issued according to multiple mechanisms, including *DNS retry*, *recursion desired flag*, *cache*, *negative caching*, and *multiple egresses*. Through extensive testing and code analysis, we have detailed these techniques used by four mainstream resolver software in Table 2, including BIND [17], Unbound [29], Knot Resolver [18], and PowerDNS Recursor [43] (also used by [30, 32, 33]). The result shows significant inconsistencies among resolver implementations, which provide potential exploitation opportunities for TsuKing.

Table 2: DNS query mechanism details of four mainstream software by default.

Software	Version	BAF related[1]		PAF related[2]			Retry		Recursion Desired (RD) flag				Cache	
		ANY qry.	Max. res. size (B)	Max. CNAME[3]	# Max. NS[4]	# Max. qry.[5]	# Count	Time-out (s)	Recursive		Forwarder		Cache	Neg.[10]
									RD=0[6]	Sent[7]	RD=0	Sent		
BIND9	9.18.10	✓	1,232	17	5	100	13	10	honor	RD=0	honor	RD=1	✓	✓
Unbound	1.17.1	✓	4,096	12	32	>100	9	>10	honor	RD=0	ignore[8]	RD=1	✓	✓
Knot	5.5.3	✗	1,232	13	5	100	3	1.2	refuse	RD=0	refuse	RD=1	✓	✓
PowerDNS	4.8.0	✓	1,232	12	13	100	1	1.5	honor	RD=0	ignore	RD=1[9]	✓	✓

[1]: Mechanisms related to bandwidth amplification factor (BAF). [2]: Mechanisms related to packet amplification factor (PAF). ✓: Enabled. ✗: Not enabled.

[3]: The maximum length of CNAME chain chasing. [4]: The maximum number of NS records that will be used. [5]: The maximum packet number sent in for resolution.

[6]: Client queries with RD=0. [7]: Queries sent to upstream servers. [8]: When allow_snoop is enabled. [9]: When forward-zones-recurse is used. [10]: Negative caching.

DNS retry, multiple egresses, and negative caching. A variety of reasons can cause DNS queries to fail, e.g., packet loss, lacking IPv6 support, and volatile infrastructure [53]. As a result, to reliably find answers, resolvers are typically designed to perform a modest number of retries when errors occur. As listed in the column "*Retry*" of Table 2, all tested DNS software will retry for no more than 13 times, should one DNS query fail. Particularly, when a DRS has multiple egresses, it may utilize different egress servers for retries of the same query. Each egress server maintains its cache independently, and on retries, it may query different authoritative servers. Security incidents demonstrate the potential DDoS attacks imposed by excessive retries. For example, during the 2021 Facebook service outage, Google Public DNS was overloaded by DNS query retries from regular clients and downstream servers [47]. A recent study proposed the TsuNAME DDoS attack model, which also works by aggregating DNS retries when domains have cyclic dependencies [39]. In the TsuKing model, we demonstrate that even a simple retry policy can result in significant attacks when exploited in conjunction with other techniques, as explained below.

Negative caching [11] provides an option for resolvers to reduce retries. Upon receiving answers showing DNS errors, resolvers with negative caching are expected to record errors, and are thus able to reply immediately when repeated queries arrive, instead of querying upstream servers again. However, while it became standard in year 1998, we find negative caching is not yet implemented in a wide range of DRSes (see Section 5).

Recursion Desired (RD) flag and cache. The RD flag in the header of the DNS query message is defined by the initial DNS standard [2]. When the RD flag is set in outgoing queries, stub resolvers or DNS forwarders ask their upstream servers to perform recursive queries and reply with final answers. If the RD flag is not set in received queries, resolvers should attempt to find answers locally, from their cache or local DNS data. Nonetheless, both previous research [59] and our findings in Section 5 indicate that a significant proportion of DNS implementations disregard the RD flag and will forward DNS queries to other servers if cache-missing.

3.2 Threat Model

TsuKing's threat model is shown in Figure 2. First, we assume the attacker can query vulnerable DRSes for his or her own domains. For open DRSes, querying from any source IP is allowed. Therefore, attackers can evade tracking by spoofing the source address. Second,
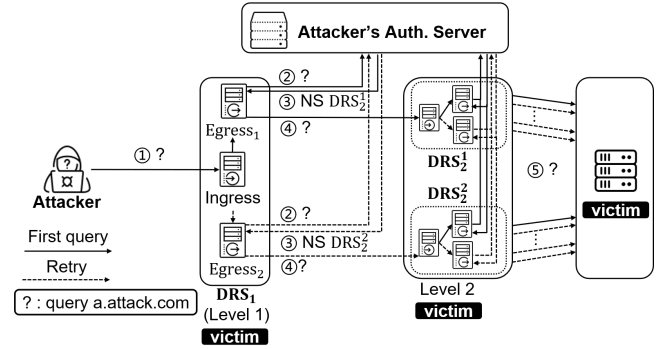

Figure 2: Threat model of basic TsuKing. The attacker's authoritative server can be eliminated in advanced attacks.

the attacker must control an authoritative server in order to return deliberately constructed replies. It is possible to register them with false identities or rent from legitimate users. Third, excluding those authoritative servers, and DRSes, which do not pose a problem, TsuKing can be used to attack any IP address.

In general, the objective of TsuKing is to provoke the victim into generating a greater number of queries (DRSes) or receiving more requests (servers), which can ultimately result in packet amplification attacks. *The key point of TsuKing is to coordinate DRSes with a small PAF to create potent amplifiers.* For DRSes with large PAFs, we show in Section 6 that attackers can exploit them directly to launch attacks. In the TsuKing attack model, the DRSes with a small PAF are carefully coordinated by the adversary, and the PAF is continuously amplified during the forwarding of queries between DRSes to eventually form a huge amplification attack. To achieve this goal, TsuKing has to overcome two challenges: *(i)* how to make DRSes work together to forward queries (the *King* technique) and *(ii)* how to transform DRSes into an amplifier to increase PAF during forwarding (the *Tsu* technique).

3.3 Attack Overview

The potential issues with DNS queries and cache, discussed in Section 3.1, are well-known to the security community [39, 44, 59] and *cannot cause a significant impact towards the Internet per se.* Nevertheless, we demonstrate that by deliberately combining these vulnerabilities, powerful amplification attacks can be constructed.

Where should queries go? King. Gummadi et al. in 2002 proposed *King* [24] to measure the latency between end hosts by guiding resolvers to use specified nameservers. To make DRSes function together, attackers also must instruct each DRS where to send its queries. Standard resolution involves obtaining referral responses from upper nameservers and querying subsequent nameservers following NS records.

Here, we propose using "maliciously" crafted NS records and incorrect handling of the RD flag to establish a new DNS resolution process by connecting two DRSes.

As shown in Figure 2, the attacker sends a query for his or her own domain (e.g., `a.attack.com`) to the vulnerable DRS_1 that has 1 ingress IP and 2 egress IPs (step ①). DRS_1 first follows the standard resolution process to resolve `a.attack.com` by receiving referrals from root and TLD nameservers, and then queries the attacker's nameserver for final answers (step ②). In lieu of an answer response, attackers reply with a new referral by providing another resolver such as DRS_2^1 as the next nameserver (step ③). Egress$_1$ of DRS_1 will untimely reach DRS_2^1 and send new queries, same to Egress$_2$ to DRS_2^2. If the query from DRS_1 with RD=0 is ignored by DRS_2^1, a new resolution process is created with DRS_1 as the query client and DRS_2^1 as the query server (step ④).

How many queries are there? Tsu. By coordinating two DRSes, attackers can exploit them to launch a joint attack against the victim by simply sending queries to the first one. However, to amplify the effect of the attack, attackers need to increase the number of generated queries. Here, we leverage common resolution mechanisms including DNS retry and multiple egresses to achieve this goal. The inspiration behind the name "Tsu" comes from the word "Tsunami", taking reference from TsuNAME [39].

According to Figure 2, DRS_1, DRS_2^1, and DRS_2^2 each have 2 egresses. After receiving requests from DRSes on the authoritative server, attackers return crafted referrals with different next nameservers for each egress. For example, attackers force 2 egresses of DRS_1 to query DRS_2^1 and DRS_2^2 separately (step ④), whilst all egresses of $DRS_2^{1/2}$ attack the victim (step ⑤). As a result, with retry from each DRS and egress, the entire resolution system becomes a potent amplifier, with massive queries flooding the victim.

Attack variants and effect. With the *Tsu* and *King* techniques, TsuKing has three attack variants, each of which could induce an impactful amplification attack targeting a variety of victims. We will describe each variant in detail in Section 4.

(i) The DNSRETRY attack coordinates a huge number of DRSes with aggressive retries to handle crafted queries from attackers, overloading the victims with high volume of DNS requests.

(ii) The DNSCHAIN attack organizes vulnerable DRSes into a resolution chain with many levels, and finally direct all duplicated DNS queries to the target victim at the final level.

(iii) Based on DNSCHAIN, the DNSLOOP attack creates a resolution loop by connecting the first and last level of a resolution chain. As the attacker continues to inject new requests into the loop, the vulnerable DRSes within the loop become progressively overloaded by the forwarded queries.

It should be noted that, although we only presented three attack variants in the paper, attackers can create more diverse attacks by changing the TTL, referral, and other techniques.

AA flag. In the TsuKing attack, DRS_1 may receive responses from DRS_2 without the AA (Authoritative Answer) flag set, even though DRS_1 expects to receive the response from the authoritative server. While RFCs [1, 2, 5, 21] specify the usage and significance of the AA flag, they do not address how to handle cases without AA. In our testing, mainstream DNS software and service providers accepted responses without the AA flag. In BIND9, responses with the AA flag set are given a higher priority [32]. Therefore, the inconsistency of the AA flag does not affect the TsuKing attack.

3.4 Vulnerabilities Exploited by TsuKing

Based on the TsuKing model and considering the requirements of three attack variants, there are two key vulnerabilities in DRSes exploited by TsuKing to conduct attacks in this paper, including recursion when RD=0 (*V1*) and no negative caching (*V2*). DRSes with V1 and V2 can be regarded as vulnerable to the TsuKing threat model, which allows attackers to continuously exploit them for one or more of the three attack variants.

(i) V1: Recursion when RD=0. DRSes in DNSCHAIN and DNSLOOP attacks need to perform recursive resolution for queries with RD=0. Otherwise, DRSes will not send any queries to the next DRS.

(ii) V2: No negative caching. If the DRS caches the negative response (e.g., timeout or failure), it will stop issuing requests for a fixed time, thus aborting the attack.

Furthermore, attackers need to consider efficiency factors when organizing and launching the three variants of attacks, which are the three characteristics of DRS below. The more pronounced (or aggressive) these three characteristics are, the easier it will be for attackers to exploit them.

(i) Retry count. The number of retries impacts the effectiveness of DNSRETRY attack, as well as the efficiency of DNSCHAIN and DNSLOOP attacks. To successfully execute a DNSCHAIN attack, the minimum value required is 2.

(ii) Retry duration. The stability of the DNSLOOP attack will increase as the retry duration increases.

(iii) Multiple egresses. The number of egresses utilized by DRSes correlates significantly with the effectiveness of the DNSCHAIN attack. To successfully execute a DNSCHAIN attack, the minimum value required is 2.

3.5 Practical Considerations

Due to the involvement of attackers' authoritative server in TsuKing, we have proposed a method to minimize attack costs. In addition, we have designed a grouping method based on egresses for each DRS to request different targets.

DRS grouping. In order to return crafted NS records to different DRSes, attackers need to know the source of DNS requests. Creating a comprehensive map of all ingresses and egresses can be challenging. When a forwarder (ingress) utilizes a large public DNS resolver as its upstream, a high number of egresses may be serving it. These egresses will also serve other ingresses that use the same public resolver. In this paper, we use a clustering strategy based on the ASO (AS Organization) information of the egress. By arranging the DRSes in accordance with the ASO, the attacker can identify which level of DRS sends the request and then respond with a specific NS record to redirect the request to a DRS in the next level.

Cache warming. In our experiment, we find that several DRSes would consult the attacker's authoritative server upon a resolving failure to validate NS records' correctness, which could increase the attack's cost. We recommend warming DRSes' cache in advance. Specifically, before launching an amplification attack, attackers query candidate DRSes and make them cache crafted NS records with a large TTL. Then attackers close the authoritative server and remove NS records in upper zones and launch the attack. These DRSes will operate as per the instructions provided by cached NS data to execute the attack. The cache warming process takes only a few minutes, and according to Xiang et al. [30], the maximum TTL of the majority of resolvers can be greater than 6 hours, or even more than 1 week. Consequently, the cost of the attack can be greatly decreased by reducing queries sent to the attacker's nameserver.

It should be noted that while cache warming can greatly alleviate the burden on attackers, it may also have an impact on the efficiency of the attack. When the attacker's authoritative server is revoked and some DRSes call on new egresses or no longer trust the current NS records, those DRSes will be unable to participate in the attack, resulting in a decrease in attack efficiency.

In Section 6.2, we showed how this approach works and that its influence on efficiency is within reasonable bounds.

3.6 Comparison With Prior Attacks

The threat model is the most fundamental difference between TsuKing and previous amplification attacks based on DNS. All of the attacks mentioned in Section 2.2 are amplified in an attempt to elicit a large or more packets from a single resolver. However, TsuKing does not emphasize the amplification capability of a single DRS. By delicately coordinating individual resolvers with limited amplification capabilities, TsuKing transforms all involved DRSes into a powerful amplifier that repeatedly sends queries to the victim.

Therefore, although TsuKing also requires the use of malicious NS records to carry out attacks, its approach is different from that of previous attacks, such as TsuNAME. In TsuKing attacks, NS records must be returned differently to various requesters so that the requests can be forwarded between various DRSes, amplifying the number of queries.

Moreover, the mode of launching an attack has evolved from the conventional approach of repeatedly requesting one or more authoritative servers via DNS recursive resolvers. Instead, the amplification of requests occurs continuously as they are forwarded between different DRSes. With this, TsuKing can launch a DDoS attack with only two devices (requesting machine and authoritative server), making the attack cost much lower than traditional DDoS attacks that rely on botnets.

In addition, unlike prior attacks, TsuKing is not the result of discrete flaws or vulnerabilities. While multiple egresses and mishandling of RD flag are not typically vulnerable per se, we demonstrate that they can serve as a potent amplifier when combined with crafted NS records together.

4 THREE VARIANTS OF TSUKING ATTACKS

In this section, we describe in detail how we craft three attack variants of TsuKing and present experiment results and measurement findings related to these attacks in Section 6.
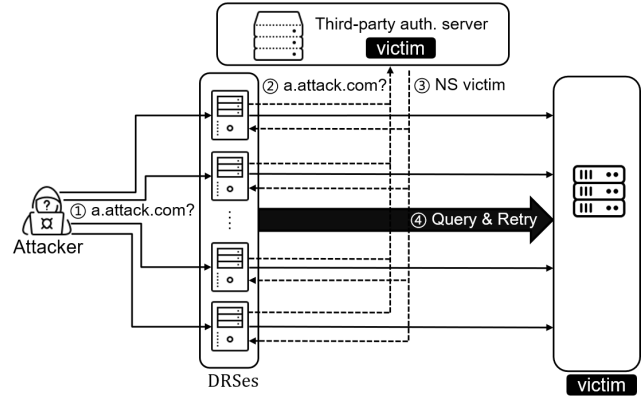

Figure 3: The DNSRetry attack.

4.1 Attack Variant 1: DNSRetry

Based on our measurements, we observed that several DRSes exhibit aggressive retry behaviors, which involving sending numerous outgoing queries over an extended period of time in order to resolve a single query. Leveraging these DRSes, therefore, attackers could launch the amplification attack directly without the need to create complicated attack paths.

4.1.1 Attack overview. Since aggressive retries of these DRSes persist for a period of time, an attacker only needs to send a new query before the DRSes' retries expire to force the DRSes to continue sending queries to the victim. Specifically, as shown in Figure 3, the attacker configures a malicious NS record on a third-party authoritative server, pointing the nameserver IP of a controlled domain to a victim. Many DNS hosting vendors, such as Cloudflare [16] and Akamai [8], provide free authoritative servers, attackers could apply for one of them. Such a malicious NS record can even have a high reputation since most hosting vendors do not verify the ownership of a domain according to [56]. The attacker then sends queries to these vulnerable DRSes, causing them to continuously flood the authoritative server and the victim.

It is worth noting that DNSRetry is a highly simplified attack technique that can still be effective even if the DRS follows the RD flag specifications and will not send outgoing queries when RD=0, as long as its retry behavior is excessively aggressive. Besides, under DNSRetry, attackers can launch attacks against vulnerable third-party authoritative servers by setting them as the victims or hosting NS records on them.

4.1.2 Attack effect. The DNSRetry *attack does not need to generate dynamic* NS *responses.* Attackers can thus deploy the NS record on any authoritative server they control, including the third-party servers. As shown in Section 6, by setting a small TTL value for the NS record, the authoritative server would receive a large number of NS queries from DRSes, making it one of the victims of DNSRetry. The other victim could be any IP that is unresponsive to DNS queries pointed by the NS record. If the victim IP is an authoritative server, it will answer requests with the NXDOMAIN rcode. DRSes will cache this response and stop sending further requests, thereby interrupting the DNSRetry attack.
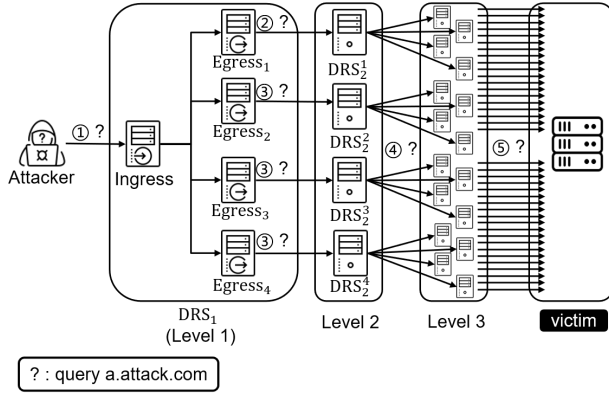
Figure 4: The DNSChain attack.

Figure 5: The DNSLoop attack.

4.2 Attack Variant 2: DNSChain

The basic idea of the DNSChain attack is to coordinate DRSes with a limited amplification capability into a new DNS resolution chain and exponentially increase the amplification factor level by level. The DNSChain attack demonstrates the effectiveness of TsuKing's two key attack techniques: *Tsu* (how to enhance the amplification effect) and *King* (how to construct a chain).

4.2.1 Attack overview. Before launching the DNSChain attack, attackers need to complete two preparations explained in Section 3.5: *DNSes grouping* and *cache warming*. Through grouping vulnerable DRSes into different levels, each level in the chain can serve as an amplifier. Subsequently, attackers inject carefully crafted NS records into each level on the controlled authoritative server, which point the next nameserver of each egress to different ingresses in the next level. In addition, after warming the cache, attackers could withdraw the nameserver and start the DNSChain attack by continuously sending queries to the first level.

As shown in Figure 4, requests are amplified within the chain level by level through several processes: *(i)* When DRS_1 receives a malicious request from the attacker, due to the malicious NS record received during the cache warming process, DRS_1 initiates a query to DRS_2^1. *(ii)* As it cannot obtain the correct response, DRS_1 begins retrying by invoking different egresses. Each egress initiates a sequence of queries towards different DRSes, as determined by the NS records obtained earlier. *(iii)* The four DRSes at L2 receive requests from DRS_1 and repeat the activity of DRS_1, causing additional DRSes at L3 to receive queries.

As the number of levels increases, an increasing number of DRSes at each level start processing queries, resulting in a higher volume of requests. At the end of the query chain, the victim will be exposed to numerous queries from the upper level DRSes, resulting in a powerful amplification attack.

4.2.2 Attack effect. The attacker utilizes multiple vulnerable DRSes to create a series of queries that increase in volume at each level of the chain, resulting in amplification of the requests. The former DRS repeats the query x times, but after the amplification process, the n DRSes at the later level raise the total number of queries to $n \cdot x$. The victim of DNSChain cannot be an authoritative server either for the same reason as DNSRetry.
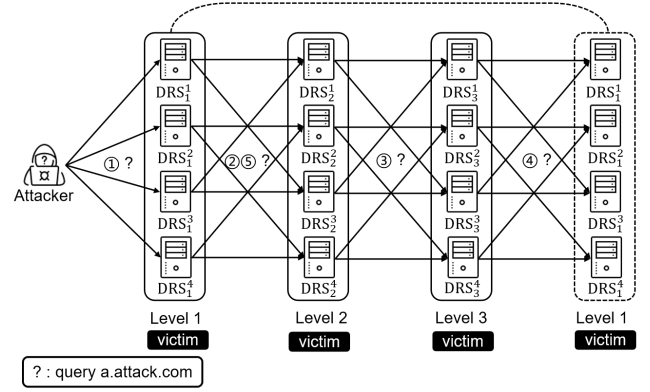
4.2.3 Attack tactics. To enhance the amplification effect, multiple strategies should be utilized to coordinate these vulnerable DRSes: *(i)* Since attackers craft a NS response based on the egress' ASO information, DRSes with the same egress-ASO should be placed at the same level; *(ii)* The amplification effect is incremental, which means that more DRSes should be present at higher levels of the query chain; *(iii)* To maximize amplification efficiency, the number of retries and egresses should be as high as possible.

4.3 Attack Variant 3: DNSLoop

The DNSLoop attack is similar to DNSChain. Attackers connect the first and last level of the DNSChain to create a loop in which requests are continuously forwarded.

4.3.1 Attack overview. This attack is executed similarly to the chain attack, except that when responding to the last level of DRSes, the NS record delegates the query to the first level of DRSes. When an attacker sends a query to any of the DRSes inside the loop, it triggers all DRSes within the loop to repeatedly process the query.

4.3.2 Attack effect. DRSes in the loop are victims since they consistently processes queries. When attackers constantly sends queries into the loop, DRSes get overwhelmed by these requests and finally reach the maximum processing capacity, resulting in DoS.

4.3.3 Attack tactics. In a DNSLoop attack, queries are used to form loops between vulnerable DRSes, causing the DRSes to attempt to process the queries continuously. The key point is to maintain the request in the loop for as long and stable as possible. We discuss two methods to achieve this. As shown in Figure 5, first, a similar number of DRSes can be placed at each level to evenly distribute the load and prevent any single level from entering a DoS state and disrupting the loop. Another factor to consider is the same request merging [49]. The DRS will wait for a response to process a request for a period of time, during which it will not issue a new request for the same query. If the waiting time is too long, the DRS will only send one request to the next level and disregard all others, terminating the loop. Hence, it is important that the duration of DRS retries is adequately long to enhance the possibility of the subsequent level of DRSes initiating new requests to sustain the loop. We will show our results to support this attack in Section 6.

Table 3: Open DRSes distribution by ingresses (top 10).

Region	Number	%	ASN	Number	%
China	444,786	33.5	4134	145,540	6.2
USA	101,669	7.7	4837	67,735	2.9
South Korea	82,997	6.2	4766	54,955	2.3
Russia	76,865	5.8	4808	48,019	2.0
Indonesia	50,884	3.8	4538	26,600	1.1
Bangladesh	42,889	3.2	9808	23,572	1.0
India	42,769	3.2	45090	22,097	0.9
Brazil	41,348	3.1	4812	19,967	0.8
France	22,851	1.7	4847	18,902	0.8
Ukraine	20,756	1.5	12389	14,922	0.6
Total 230 regions			**Total 23,626 ASes**		

5 FINDING VULNERABLE DRSES

In this part, first, we undertake a network scan to discover potential DRSes with basic ethical considerations. Then we evaluate vulnerabilities given in Section 3.4 and analyze vulnerable DRSes. We conclude with a summary of potential DRSes that can be exploited to conduct the TsuKing attack.

5.1 Collecting Open DRSes

Open DRSes list. To gather open DRSes, we utilize *XMap* [31] to query domains on UDP port 53 over the IPv4 address space. The authoritative servers for queried domains are under our control, allowing us to identify the ingress and egress of the open DRSes. We identify the IP addresses that returned the correct DNS answers as open DRSes. We perform the scans several times with a limited scanning rate and obtained 1,326,499 open DRSes covering 230 countries (or regions) and 23,626 ASes.

Distribution of DRSes. Using the GeoLite2 database [35], we examine the geographical and AS information of all DRSes and listed the top ten areas and ASNs in Table 3. The distribution of open DNS resolvers may create a bias, wherein only a few regions are responsible for the majority of these resolvers. However, our focus is on the prevalence of threats, not their distribution features.

5.2 Vulnerability Analysis of DRSes

To identify vulnerable DRSes, we conduct several measurements on discovered 1.3 million open DRSes. This enables us to identify DRSes with specific characteristics as mentioned in Section 3.4.

5.2.1 RD flag. A DRS that ignores the RD flag can participate in DNSChain and DNSLoop attacks, by receiving and forwarding requests. We aim to spot DRSes that still perform recursive queries when receiving a request that unsets the RD flag (RD=0).

Measurement. We query these 1.3 million open DRSes for a special subdomain that encodes the IP address of the DRS in decimal form. For example, the domain name for a query sent to 8.8.8.8 is `134744072.attacker.com`. Since none of the requests have the RD flag set, we can gather access records from our authoritative servers and use the request information to identify which DRS is experiencing issues and match their ingresses and egresses.
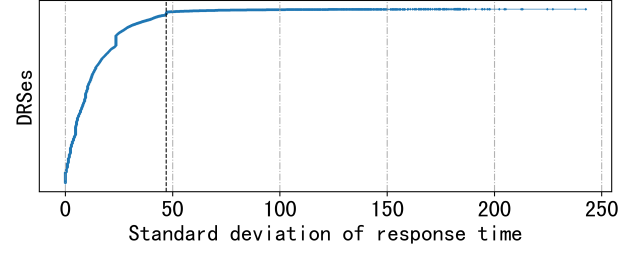
Figure 6: Distribution of Std. Dev. of DRS response time (each point represents a DRS).

Results. Overall, **361,621 (27.26%)** DRSes continue to perform recursive resolution when they encounter requests that unset the RD flag, indicating that this issue is fairly prevalent.

5.2.2 Negative caching. DRSes with negative caching are able to record the resolution failure and respond instantly based on the cache when the same query is received again. According to our analysis in Table 2 and testing, resolvers cache the nameserver IP and refuse to process subsequent requests, which could interrupt DNSChain or DNSLoop. Although the TTL of the negative cache may not be a large value, an attacker can construct multiple query chains and launch attacks separately to circumvent this problem. Nonetheless, this renders the attack operation more intricate. Therefore, we choose DRSes without negative caching to perform the DNSChain and DNSLoop attack. For DNSRetry attacks, since a DRS can reach an extremely high number of retries indicating that there is no negative caching, thus for the DNSRetry attack, we decide whether a DRS is vulnerable just by the retry count.

To ascertain whether a DRS has negative caching, we analyze the difference in the response time that the DRS exhibits for several consecutive failed requests. If a DRS has negative caching, it will cache failed results, such as timeout, so that when it receives the same query again, it can respond promptly. Therefore, if there is a significant difference in the response time for consecutive requests, we can conclude that the DRS has negative caching.

Measurement. We query a subdomain within our domain to each DRS, which also encodes its IP address in the domain. Upon receiving a response, we promptly re-issue the query, repeating this process 4 additional times (for a total of 5 queries). By analyzing the change in the response time for the 5 queries, we can infer whether the DRS has negative caching. Our authoritative server will not respond, resulting in a timeout error for DRSes.

Result. We exclude non-robust results, especially those with the response times less than *500ms* for all five queries, as such rapid responses are unlikely to be indicative of meaningful query results. Finally, we obtain 594,174 useful results, and use the *Standard Deviation* of 5 response times (0.1s unit) as the criterion for evaluation.

Figure 6 presents a scatter plot of all results. Each point represents a DRS and the x-axis represents its standard deviation. Results show that the change in standard deviation starts to increase substantially at a value of 47, thus we use this value as the threshold to determine whether a DRS has negative caching. Consequently, there are 576,967 DRSes without negative caching, accounting for 97% of our valid results and approximately 43% of all open DRSes.

Table 4: The retry distribution of open DRSes.

Number of retries	Count	Proportion
> 2	925,500	69.8%
> 10	407,581	30.7%
> 100	31,660	2.4%
> 1000	529	0.04%
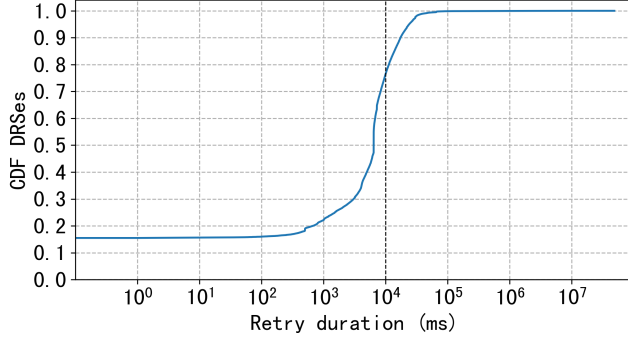
Figure 7: CDF DRSes by retry duration.

5.2.3 Retry count. For the DNSRᴇᴛʀʏ and DNSCʜᴀɪɴ attacks, the attack becomes more efficient and powerful as more retries of the DRSes are performed. By exploring the retry count of the DRS, an attacker can conduct attacks with vulnerable DRSes with the high amount of retries.

Measurement. We send 10 distinct queries to each DRS and refrain from providing any response on the authoritative server. Instead, we record the pertinent request information. Due to timeout, DRSes will retry, therefore by counting the number of requests, we can determine the retry status of each DRS.

Result. As shown in Table 4, DRSes with retry counts of over 100 account for 2.4%, and those with retry counts of over 10 account for 30.7%. The highest retry count even reached *117,541*. This demonstrates that retrying is a very common resolution operation for DRSes and that a large number of DRSes exhibit extremely aggressive retry behaviors.

5.2.4 Retry duration. A lengthy retry time can be leveraged to prevent request merging. During a DNSLᴏᴏᴘ attack, if a DRS merges identical requests and subsequently receives the same request again within the loop, it may cause premature termination of the loop by failing to forward the request further. However, if the retry duration of a DRS in the loop is so long that it exceeds the processing wait time of DRSes that merge requests, the loop will be constructed again, ensuring the DNSLᴏᴏᴘ's stability.

Measurement. We reuse the results from the *DNS Retry* measurement and calculate the average time consumed by repeated requests during the retry process.

Result. Figure 7 shows that 76.6% of DRSes require less than 10 seconds to process a query that receives no authoritative response from our controlled authoritative server. Among these, approximately 26.7% of DRSes take 2 seconds or less.

Table 5: fpdns measurement results of vulnerable DRSes.

DNS software	Count	Percentage[1]
Mikrotik dsl/cable	50,277	79.9%
Microsoft Windows DNS 2000	4,631	7.3%
Paul Rombouts pdnsd	3,420	5.4%
Raiden DNSD	1,272	2.0%
Meilof Veeningen Posadis	975	1.5%

[1]: The total number is 62,919 DRSes with identified fpdns results.

Furthermore, we have observed that among all DRSes that are non-compliant with the RD flag regulation, some may experience a prolonged retry duration of up to *48,496* seconds. This indicates that an attacker can leverage these DRSes with exceptionally long retry durations to participate in the DNSLᴏᴏᴘ attack to ensure the attack's stability. Even without such DRSes, our experiments will show in Section 6 that a resolver retry length of at least 2s is sufficient enough to ensure the stability.

5.2.5 Multiple egresses. The more egresses a DRS has, the better its ability to trigger extra DRS units in the subsequent tier. We can use DRSes with multiple egresses to conduct attacks, resulting in a high impact.

Measurement. We analyzed the historical access data of all DRSes to the authoritative server and counted the number of egresses and the number of ASOs to which they belong.

Result. According to the results, 52% of the DRSes have more than 10 egresses, among which 30.7% of the DRSes have egresses belonging to 2 or more ASOs.

5.3 Vulnerability Assessment of DRSes

As defined in Section 3.4 of the TsuKɪɴɢ threat model, a DRS is considered vulnerable if there is erroneous handling of the RD flag and no negative caching is implemented. To gain a comprehensive understanding of the existence of these vulnerabilities, we utilize fpdns that is a common DNS software fingerprint identifying tool in the DNS community [13] to measure vulnerable DRS software.

Out of all open DRSes, **14.5%** or **191,694** were discovered to be vulnerable. By performing fpdns measurements on these 191,694 vulnerable open DRSes, we obtained results from 62,919 and listed the top 5 DNS software in Table 5. Mikrotik [36] accounted for nearly 80% of them. In Section 7.2, we conducted further analysis on RouterOS [37] (the router operating system and software used by Mikrotik) and identified the underlying cause of vulnerabilities.

On the other hand, as discussed in Section 3.4, there are three key characteristics that have a significant impact on the attack efficiency. DRSes exhibiting these characteristics are more susceptible to being targeted by attackers or being exploited to launch the TsuKɪɴɢ attack. We summarized the thresholds for these vulnerable characteristics of the three attack variants and presented the measurement results in Table 6.

From our measurements and analysis, we found a large scale of open DRSes and demonstrated that there are a significant number of DRSes that can be utilized to conduct TsuKɪɴɢ. In the next section, we will show our real-world evaluation of TsuKɪɴɢ, whose threat can never be underestimated.

Table 6: Vulnerability analysis results of open DRSes.

Threat scenarios	Ignoring RD=0	No negative cache	Retry >10x	Retry >1,000x	Retry duration exceeding 2s	Multiple egresses	Count	Percentage
TsuKing	✓	✓	-	-	-	-	191,694	14.5%
DNSRetry	-	✓	-	✓	-	-	529	0.04%
DNSChain	✓	✓	✓	✓	-	✓	62,644	4.7%
DNSLoop	✓	✓	-	-	✓	-	154,884	11.7%

6 EVALUATION OF THREE VARIANTS

In this section, we evaluate three attack variants in the real-world experiments to demonstrate their feasibility and impacts. Based on previous measurement results, we select appropriate DRSes for the attack. With several small-scale controlled experiments under ethical considerations, we show that the packet amplification factor (PAF) can be more than *3,700*, which is just a lower bound.

6.1 Evaluation Experiment 1: DNSRetry

Retry improves the resolution process, but also raises the server's burden. Aggressive retry furthermore introduces the DNSRetry attack. Under DNSRetry, "malicious" DNS requests flood the victim and third-party authoritative server. To measure the effect of DNSRetry, we conducted the following controlled experiments.

6.1.1 Experiment design. In our experiments, three servers are deployed. The first server acts as the attacker and continuously sends requests to the vulnerable DRSes. The second server functions as an authoritative server, providing malicious NS responses to delegate the request to the third server (victim). Both the authoritative server and the third server are targets of the attack. To evaluate the attack on the authoritative server, we have set the TTL value of the NS response to *1s*. We have selected ten of the DRSes that perform over 1,000 retries and regularly send A-type DNS queries to them based on the request duration of each DRS.
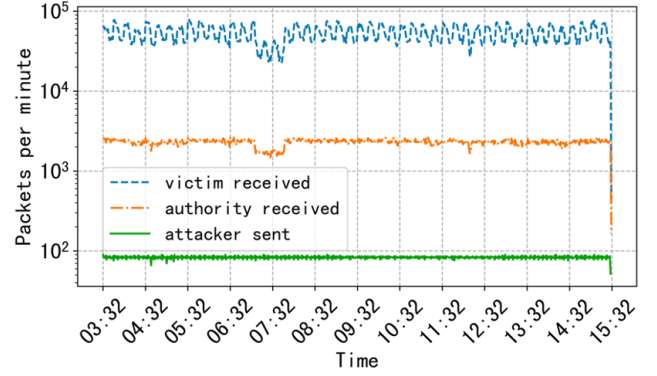
By counting the number of sent and received DNS packets on the three servers, we can visualize the effect of the DNSRetry attack. For simplicity, we only consider the DNS queries of type A, as was done in the DNSChain and DNSLoop experiments.

6.1.2 Results. Figure 8 shows the result of the DNSRetry experiment. The experiment lasted about 12 hours, and both the authoritative server and victim continued to receive massive requests.

Throughout the duration of the experiment, the attacker's sending rate an average of 1.38 p/s (packets per second). The packet receiving rate of the authoritative server was an average of 38.4 p/s, whereas the victim was an average of 882.6 p/s. The amplification factor for the authoritative server reached 27.7× and 638× for the victim. If the attacker coordinates more DRSes and increases the sending rate, the result can be much more devastating.

6.2 Evaluation Experiment 2: DNSChain

In this part, we evaluate the feasibility of the DNSChain attack using vulnerable DRSes identified in Section 5.2. We also examine the efficiency of cache warming and analyze the differences between the attack in theory and real-world.

Figure 8: The number of packets sent and received in the DNSRetry experiment.

The amplification factor of DNSChain can be calculated in theory by analyzing the number of retries for each DRS. However, the actual value may not match the expected one due to the complexity of the network and the DRSes (e.g., the same egress being used for multiple retries). Thus, we compare the actual attack results to the expected ones, or the *decay rate* (DR), which we define as:

$$DR = 1 - \frac{victim\ received\ requests\ actually}{victim\ received\ requests\ theoretically} \quad (3)$$

6.2.1 Theoretical amplification. For convenience, we name DRSes based on their levels. For example, the DRS at level 1 is named R_1, and the three DRSes at level 2 are labeled R_2^1 to R_2^3. "N" denotes the number of retries for the resolver, and the number of retries for R_1 is N_{R_1}. For *Experiment 3-2*, we can derive the formula for calculating the amplification factor (same to other experiments):

$$\frac{\frac{N_{R_1}}{3} \times (N_{R_2^1} + N_{R_2^2} + N_{R_2^3})}{6} \times (N_{R_{last}^1} + \cdots + N_{R_{last}^6}) \quad (4)$$

6.2.2 Experiment design. We organized 9 small-scale experiments to perform horizontal comparison (analyzing the number of levels) and vertical comparison (analyzing the number of DRSes).

Table 7 shows the experiment settings and results. The experiments are divided into three groups, with query chains of length 3, 5, and 7, respectively. DRSes that are used in small-scale experiments will also participate in larger-scale experiments. For example, the 4 DRSes in the "level last" category of *Experiment 3-1* are also utilized in the subsequent 8 experiments. Our authoritative server responds to the DRSes in the "level last" level with an NS record that points to the victim, which is one of our controlled hosts.

Table 7: Experimental results of DNSChain.

| No.	# of DRSes							# Actual victim recv. p/s[*]	# Theoretical victim recv. p/s[*]	Actual PAF	Theoretical PAF	Decay rate
	Level1	Level2	Level3	Level4	Level5	Level6	Level last					
3-1	1	2	-	-	-	-	4	55.6	2,951.1	167	8,853	0.981154986
3-2	1	3	-	-	-	-	6	61.2	2745.1	184	8,235	0.977677665
3-3	1	4	-	-	-	-	8	96.1	2,515.1	288	7545	0.961797415
5-1	1	2	4	8	-	-	16	120.9	194,410.7	362	583,232	0.999378140
5-2	1	3	6	12	-	-	24	143.3	253,242.3	430	759,727	0.999434197
5-3	1	4	8	16	-	-	32	196.9	227,112.7	591	681,338	0.999132899
7-1	1	2	4	8	16	32	64	705.2	260,761,956.3	2,116	782,285,869	0.999997296
7-2	1	3	6	12	24	48	96	1,036.8	361,085,125.9	3,110	1,083,255,378	0.999997129
7-3	1	4	8	16	32	64	128	1,234.1	305,539,248.3	3,702	916,617,745	0.999995961

[*] : Packets / Second.
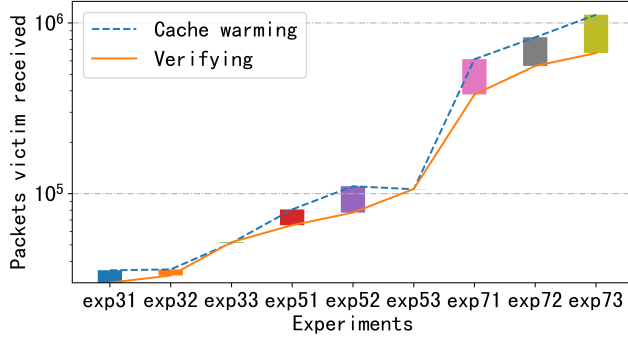
Figure 9: Packets received by the victim in different phase of the DNSChain experiments.

To demonstrate the effectiveness of our proposed cache warming method, we also conduct experiments including the following two phases. *(i) Cache warming phase.* We send *180* queries within *1* minute to warm up the cache. During this period, the authoritative server responds with crafted NS records that have a TTL of *10* minutes. *(ii) Verifying phase.* The authoritative server stops responding and we send another *180* verifying queries during *9* minutes.

6.2.3   Results. The results in Table 7 are the scaled-up results of the 180 verifying queries (verifying phase). Through this results, we can analyze the amplification of several experiments. Figure 9 shows comparison of packets received by the victim in the two phase of several experiments. Since we sent 180 requests in both phase, we can determine whether the cache warming method is feasible and the degree to which it affects the efficiency of the attack.

Amplification and decay rate. From Table 7, we conclude two main findings. *(i)* The decay rate is very severe, resulting in actual amplification that is much lower than the expected theoretical value; *(ii)* Despite this, the amplification effect through the experiments is still extremely high. For example, in *Experiment 7-3*, 253 DRSes were used to achieve an amplification factor greater than *3,700*.

The effectiveness of a DNSChain attack depends on the number of DRSes in the last level, allowing attackers to amplify the attack by stacking more DRSes as the layers increase. Additionally, attackers can take advantage of the characteristics of DRSes, such as placing DRSes with aggressive retry policies in the last level, to further increase the amplification factor of DNSChain attacks.

At the same time, the decay rate is very severe. In addition to the potential issue of some DRSes utilizing the same egresses during retry attempts, we have identified two primary contributing factors. (i) The theoretical amplification factor is the most optimistic scenario where each query sent by DRS_1 to DRS_2 will generate a new query. However, in our tests, DRSes can aggregate queries for a specific domain name during a processing cycle (same domain request merging), which means that most of the queries sent by DRS_1 to DRS_2 in the DNSChain attack are redundant. As shown in Table 7, for the same query chain length, the higher the number of DRSes in the chain, the higher amplification factors result in, but the decay rate will decrease. In other words, it is more effective to have more DRSes receive one query rather than making fewer DRSes receive multiple queries. This confirms that the aggregation operation can cause a decrease in the amplification factor. (ii) On the other hand, the data used to calculate the theoretical amplification is based on the situation where the authoritative server does not respond (timeout). In actual attacks, DRS_1 may receive a response from DRS_2, such as "Failure", which may reduce the number of retries by DRS_1 and result in a lower actual amplification than the theoretical value. We discussed this further in Section 7.3.

Cache warming. From Figure 9, we can see that the total number of packets received by the victim in the verifying phase is nearly equal to the number received in the cache warming phase. Although the number of packets received in the verifying phase is slightly less, it does not have a significant impact on the attack. The results indicate that cache warming can be well applied to TsuKing attacks.

6.2.4   Long time experiment. To observe DNSChain's stability, we performed another controlled experiment lasting 6 hours with the same experiment settings of *Experiment 5-3*.

Similar to previous experiments, the entire process consists of two phases. *(i)* For cache warming, we send 3 queries per second for 10 minutes, and return NS responses on the authoritative server, with a TTL value of *6h*. *(ii)* Then, we stop the authoritative server and reduce the rate of sending queries to 1 query per 3 seconds.

As shown in Figure 10, after losing the support of the authoritative server, the entire attack continues to exert pressure on the victim by depending on the resolver cache. Throughout the experiment, the attacker sent a total of *17,864* packets *(0.8 p/s)*, the attacker's authoritative server sent a total of *8,461* packets *(12.8 p/s)*, and the victim received a total of *4,557,336* requests *(206.4 p/s)*.
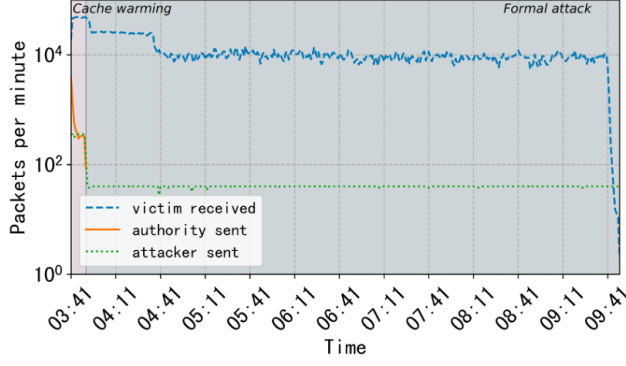
Figure 10: Long time DNSCʜᴀɪɴ experiment.

Figure 11: Results of DNSCʜᴀɪɴ simulation experiments.

6.2.5 Attack impact testing. Due to ethical considerations, we cannot organize high-load experiments in real-world networks to test the impact of DNSCʜᴀɪɴ attacks. Therefore, we built an experimental network to test the effectiveness of this attack.

In this network, we used RouterOS [37], which has the highest proportion of vulnerable DRSes in our measurements, to simulate 36 vulnerable DRSes. All devices, including the attacker and the victim, are assigned within an IPv4 subnet with a prefix length of 26. All traffic in the experiment was transmitted within the experimental network we set up, without any impact on the real-world network.

In Experiment-1, we first organized 31 RouterOS devices into a DNSCʜᴀɪɴ architecture of 3 levels (DRS_1^1, DRS_2^{1-5}, and DRS_3^{1-25}), achieving a PAF of *125*. Then, in Experiment-2, we further organized DRS_3^1 at *level3* to request DRS_4^{1-5} at level4, namely, DRS_3^{2-25} and DRS_4^{1-5} all requesting the victim, resulting in a PAF of *145*.

As shown in Figure 11, after the attack began, the attack traffic quickly increased to *14 Mb/s*. The request traffic experienced by the victim escalated swiftly in correlation with the surge in the attack traffic, reaching its zenith prior to attackers attaining their peak traffic. In Experiment-1, the victim's traffic peak was *1,080 Mb/s*, while in Experiment-2, with the further increase of the amplification factor, the victim's traffic peak reached *1,238 Mb/s*.

According to Kopp et al. [28], the average traffic volume for DNS-based DDoS attacks is around *2 Gb/s*. Based on our simulation results, it would require 78 devices organized in a 4-levels attack architecture (DRS_1^1, DRS_2^{1-5}, DRS_3^{1-25}, and DRS_4^{1-75}), creating a PAF
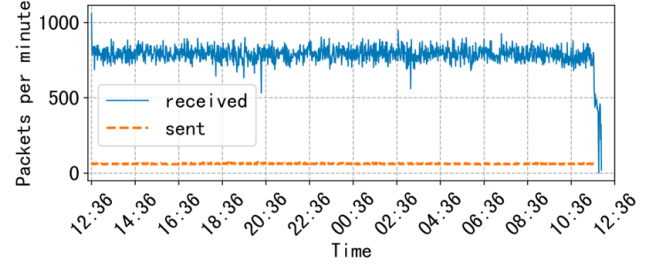

Figure 12: The number of packets received and sent by the Level 0 per minute in the DNSLᴏᴏᴘ experiment.

of 235, to achieve this level of attack. The measurement results in Section 5.3 and the small-scale real network experimental results in Section 6.2.3 suggest that attackers can launch attacks far beyond this level in real networks.

6.3 Evaluation Experiment 3: DNSLᴏᴏᴘ

In this part, we perform a 24-hour experiment to evaluate the stability of DNSLᴏᴏᴘ. We constructed a 7-level loop. To control the traffic size in the loop and stop the experiment at the appropriate time, we used a RouterOS device as the entry point of the loop.

6.3.1 Experiment design. Specifically, we set up a RouterOS host (*Level* 0) with the open DNS function enabled and the upstream to 2 DRSes at *Level* 1. We send queries at *Level* 0 at 1 p/s to control the traffic size within the entire loop. Our authoritative server responds to the *Level* 7 DRSes with NS records containing information about the RouterOS host, creating a query loop. The number of DRSes from *Level* 1 to *Level* 7 is *2, 11, 16, 15, 50, 101, and 15*.

After cache warming, we send 1 query to *Level* 0 at the beginning, and then observe and analyze packets received and sent at *Level* 0.

6.3.2 Results. We did not analyze the data of the cache warming phase. Figure 12 shows the experiment persisted for 24h and the rate of packet reception at *Level* 0 remained highly stable throughout the duration until terminating the experiment. This demonstrates DNSLᴏᴏᴘ can be launched practically in real networks and can make vulnerable DRSes process the same query forever.

On the other hand, when *Level* 0 initiates a query, it sends one packet to each of the two upstream servers. Throughout the experiment, *Level* 0 sent a total of *86,380* packets (*1 p/s*) and received *1,100,320* packets (*12.7 p/s*). This means that *Level* 0 processed the same request *43,190* times during the DNSLᴏᴏᴘ attack.

Under another experiment, we selected several DRSes with the retry duration ranging from 1 to 3 seconds to create a 3-level query loop. Results show that the loop also remains stable. Based on this, we conclude that a DRS with a request duration of at least 2 seconds can be exploited to conduct the DNSLᴏᴏᴘ attack. From Section 5.2, our measurements show that there are about 73% open DRSes with a retry duration that can reach 2 seconds or longer.

Our experiment also reveals that the DNSLᴏᴏᴘ attacks remain feasible even if the NS record information that replies to the egress of some DRSes corresponds to their own ingress (a self loop).

7 CAUSE AND IMPACT IN THE REAL WORLD

In this section, we attempt to investigate the causes of the issues that lead to the TsuKing attack and discuss the potential impact of some real-world factors on TsuKing. There are various factors that may contribute to these issues and have an impact on TsuKing. Here, we simply summarize some main findings from our study.

7.1 Retry Behavior

Our experiments show that numbers of DRSes have a significant amount of retries, which seems to contradict the processing logic of mainstream DNS software (listed in Table 2). After tracking and analyzing all the problematic DRSes with aggressive retries, we discovered that a prominent public DNS provider (360DNS) experienced over 60 retries. After receiving our report, the technical team conducted an examination and discovered that it was caused by the hierarchical structure of their DNS resolution system. The amount of retries is increased by their internal level-by-level forwarding.

We deduce that multiple factors are contributing to a large number of retries like misconfigurations, software errors, network loops, and others. However, the problem stemming from a complex and hierarchical resolution structure remains highly relevant in today's rapidly growing public DNS services.

7.2 RD Vulnerability

To identify vulnerable vendors, we evaluate both several public DNS services and our discovered DNS resolvers. For public DNS services, we select 8 providers with a majority of world use from APNIC's database [12], including Google Public DNS, Cloudflare DNS, Level3 DNS, Neustar DNS, DNSPoD Public DNS, Ali DNS, 114DNS, and 360DNS. We find that Ali DNS, 114DNS, and DNSPod Public DNS ignore the RD flag and always perform recursive queries. We communicated with technical staff from these vendors. The issue with 114DNS was due to a temporarily open special feature while testing new functionality. In the other two vendors, the RD flag was not maintained during the transmission process in the multi-layer architecture of the resolvers. So far, all three vendors have addressed and resolved the issue.

On the other hand, although Google Public DNS (GPDNS) honors the regulations of the RD flag, we still observe requests from GPDNS on our authoritative servers during the measuring of Section 5.2.1. This suggests that the issue exists in some downstream forwarders and with multiple egresses of GPDNS [44], and thus the impact is extremely amplified. Based on the results of fpdns in Section 5.3, we further studied the reasons for the vulnerability of RouterOS [37]. RouterOS is a router operating system and software used by Mikrotik that enables the DNS forwarding function. However, if receiving a query with RD=0, RouterOS will still forward it to upstream when cache missing and reset the RD flag to 1. As tested in Table 2 and Section 3.1, this issue also affects Unbound and PowerDNS Recursor in their forwarding mode.

7.3 Negative Response

In this part, we measure the effect of the negative response on the TsuKing attack. On the authoritative server, we return different types of DNS responses to our identified vulnerable resolvers to analyze their retry behaviors.
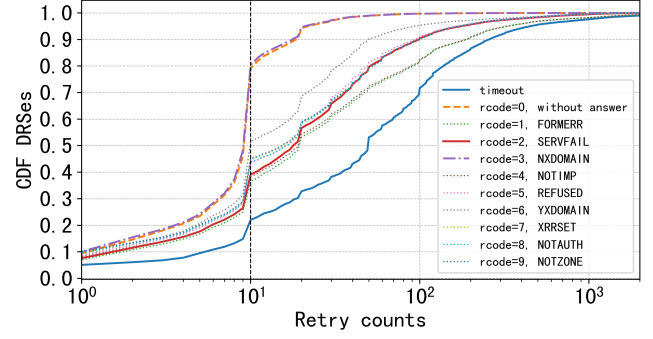

Figure 13: CDF for DRSes of retry counts in various cases.

Experiment design. In order to compare the number of retries in 11 different cases (rcode=0-9 and timeout), we query 110 different subdomains for each DRS (10 for each case to avoid packet loss), encoding the subdomains according to the different cases. Depending on the information contained within subdomains, the authoritative server responds differently, either by refraining from processing them or providing a specific type of response. We then counted the number of resolver queries for each case.

Result. We counted the number of retries for all open DRSes in various cases and plotted them in a CDF figure. As shown in Figure 13, when (i) a response with an RCODE of 0 (NOERROR) is received, even if the response does not contain an answer, or (ii) a response with an RCODE of 3 (NXDOMAIN) is received, most DRSes concur with the result and stop retrying. (iii) DRS retry behavior is particularly severe in the case of Timeout (no response from the authoritative server). (iv) Retrying is somewhat reduced when a response with an RCODE of 2 (SERVFAIL) is received.

According to Li et al [33] and Moura et al [39], when resolution fails, the processing logic of most DRSes is to either not respond or to reply with a SERVFAIL type response. This might also explain the large difference in amplification compared to the theory discussed in Section 6.2.

8 DISCUSSION

8.1 Mitigation

In order to defend against TsuKing attacks, it is necessary to pay attention to previously neglected issues. In this part, we propose four approaches to mitigate this problem.

Honoring the RD flag. The root cause of the TsuKing attack is that DRSes still perform recursive queries when they receive a request that unsets the RD flag. Hence, it is recommended that DRSes, especially forwarders, exercise caution while handling such requests. We recommend that all DRSes follow the RD policy and only answer based on the cache or local zone when they receive a query that unsets the RD flag.

Implementing negative caching. Negative caching helps to prevent DRSes from engaging in invalid retry behavior and minimize the risk of attacks. Especially for DNSLoop attacks, the negative caching can prevent the loop from running. Therefore, we recommend that DRSes implement negative caching if possible.

Avoiding aggressive retries. The amount of retries is directly related to the risk of attacks. We show that the aggressive retry itself could be exploited to amplify traffic. Thus, we recommend that the total number of requests for a query should be limited, e.g., 8 (the average number of four mainstream software in Table 2).

Optimizing egress schedule. Optimizing the management of egress schedules for DRSes is crucial to prevent a widespread attack. This can be achieved by restricting the number of egresses used for retrying a request or sharing cache, leading to a significant reduction in the impact of the DNSChain attack.

8.2 Existing Defenses

Although there is an increasing number of best practices being implemented to enhance security, such as 0x20, DNSSEC [25], DNS Cookies [4], DoT [27], DoH [26], and others, these practices primarily focus on safeguarding the privacy, integrity, and accuracy of DNS resolution, rather than preventing DNS amplification attacks. Some specific mainstream measures for preventing DNS amplification attacks are as follows. We analyze their impact on TsuKing.

(i) Refusing to respond to sensitive query types. In traditional DNS reflection attacks, defenses mainly rely on detecting abnormal requests, such as requests for special types like ANY. However, in TsuKing, the attack does not depend on the authoritative server or recursive response to generate an amplification attack, but rather on making more and more DRSes send requests through the forwarding process. Therefore, the requests made do not necessarily have to be of sensitive types like ANY.

(ii) Limiting repeated requests. Traditional reflection amplification attacks involve repeated requests to continuously generate large response packets. Resolvers or authoritative servers can detect this abnormal behavior and limit repeated requests from a single IP address. However, in TsuKing, DRSes cannot obtain and reply with the correct results. Hence, regarding repeated requests for a domain name, DRSes cannot determine whether this is malicious behavior.

(iii) Source address validation. Traditional reflection amplification attacks rely on forging source addresses to carry out the attack. Resolvers and authoritative servers can use source address validation techniques to refuse processing these requests. However, in TsuKing, the attack relies on queries rather than responses, so the attacker does not need to forge the IP address of victims.

(iv) Rate limiting. Resolvers and authoritative servers limit the request rate and outgoing transmission rate from the same IP address to curb DDoS attacks. This is an effective method for any situation attempting to use DNS devices to carry out DDoS attacks. In this case, TsuKing would also be affected. However, there are many DRSes with TsuKing vulnerabilities, and attackers can organize larger-scale attacks to bypass this impact.

In summary, we believe that TsuKing, as a new type of DNS amplification attack, has become more difficult to detect and defend against due to the differences in its attack techniques compared to previous attacks.

8.3 Ethical Considerations

Due to that our experiments include several DNS vendors and open DNS resolvers, our experiment design involves several ethical considerations to reduce the impact.

First, all experiments were conducted using our own domain for research purposes. After each experiment, we deleted the associated records on the service platforms to prevent potential exploiting.

Second, throughout our work, we attempted to restrict the number of requests sent to the DRSes. When measuring the characteristics of the DRSes, we sent only 116 requests for each DRS. In all experiments, the highest rate of sending requests was 3 queries per second. Although our experiments had a 3700× PAF, this was the result of 128 DRSes in the last level working together. Similarly, the rate of requesting *Level* 0 in our DNSLoop experiment was *12.7 p/s*, which was only *0.8 p/s* for each DRS in *Level* 7.

Third, our controlled experiments only targeted our own server, leaving other servers and underlying network infrastructure unaffected. Specifically, in Section 6.2, Experiment 7-3 yielded the maximum bandwidth, wherein we sent 180 DNS requests within 9 minutes, resulting in 666,385 packets sent to the victim server, or 200KB/s of traffic. We consider the overhead caused by the experiment modest compared to the capabilities of most servers, network switches, and routers.

In the end, we report vulnerabilities to all relevant vendors in order to guarantee responsible disclosure.

8.4 Responsible Disclosure

DNS software. We found that RouterOS, Unbound, and PowerDNS Recursor all have an issue that will forward queries with a cleared RD flag to the upstream server when the cache is missing and reset the RD flag to 1. We have reported this issue to all of them. Up to now, all of them have confirmed the issue [42, 50]. PowerDNS has submitted a new change request; however, they have yet not acknowledged that this issue could potentially result in a security concern. TsuKing has been assigned 3 new CVE numbers.

Public DNS services. We also discovered that 114DNS, AliDNS, and DNSPod all have an issue with recursive processing of queries that unset the RD flag. After reporting this to their operators, all of them confirmed the issue and fixed it.

9 CONCLUSION

In this paper, we proposed TsuKing, a new DNS amplification attack that delicately coordinates numbers of vulnerable DRSes with limited amplification capabilities into powerful DoS amplifiers with a PAF of above 3,700. TsuKing has a significant and widespread impact in the real-world, around 14.5% of 1.3M open DRSes are vulnerable, as well as several popular DNS software and public DNS services, such as Unbound, PowerDNS, and 114DNS. We conducted extensive Internet measurements and controlled experiments to demonstrate the feasibility and enormous risks of TsuKing.

ACKNOWLEDGMENTS

REFERENCES

[1] 1987. Domain names - concepts and facilities. RFC 1034. https://doi.org/10.17487/RFC1034
[2] 1987. Domain names - implementation and specification. RFC 1035. https://doi.org/10.17487/RFC1035
[3] 114DNS. 2023. 114DNS. http://www.114dns.com/about.html
[4] Donald E. Eastlake 3rd and Mark P. Andrews. 2016. Domain Name System (DNS) Cookies. RFC 7873. https://doi.org/10.17487/RFC7873
[5] Donald E. Eastlake 3rd, Eric Brunner-Williams, and Bill Manning. 2000. Domain Name System (DNS) IANA Considerations.
[6] Joe Abley, Ólafur Guðmundsson, Marek Majkowski, and Evan Hunt. 2019. Providing Minimal-Sized Responses to DNS Queries That Have QTYPE=ANY.
[7] Yehuda Afek, Anat Bremler-Barr, and Lior Shafir. 2020. NXNSAttack: Recursive DNS Inefficiencies and Vulnerabilities. In Proceedings of USENIX Security '20.
[8] Akamai. 2023. Edge DNS. https://www.akamai.com/products/edge-dns
[9] Kamal Alieyan, Mohammed M. Kadhum, Mohammed Anbar, Shafiq Ul Rehman, and Naser K. A. Alajmi. 2016. An Overview of DDoS Attacks based on DNS. In Proceedings of ICTC '16.
[10] Marios Anagnostopoulos, Georgios Kambourakis, Stefanos Gritzalis, and David K. Y. Yau. 2018. Never Say Never: Authoritative TLD Nameserver-powered DNS Amplification. In Proceedings of IFIP '18.
[11] Mark P. Andrews. 1998. Negative Caching of DNS Queries (DNS NCACHE).
[12] APNIC. 2023. Use of DNS Resolvers for World (XA). https://stats.labs.apnic.net/rvrs
[13] Roy Arends and Jakob Schlyter. 2020. fpdns. https://github.com/kirei/fpdns
[14] BIND. 2020. CVE-2020-8616. https://kb.isc.org/docs/cve-2020-8616.
[15] Jonas Bushart and Christian Rossow. 2018. DNS Unchained: Amplified Application-Layer DoS Attacks Against DNS Authoritatives. In Proceedings of RAID '18.
[16] Cloudflare. 2023. Cloudflare DNS. https://developers.cloudflare.com/dns/
[17] Internet Systems Consortium. 2023. BIND 9. https://bind9.readthedocs.io/en/v9_18_10/
[18] CZ.NIC. 2023. Knot Resolver. https://gitlab.nic.cz/knot/knot-resolver/activity
[19] CZ.NIC. 2023. Knot Resolver: policy.REFUSE. https://knot-resolver.readthedocs.io/en/stable/modules-policy.html?#policy.REFUSE.
[20] 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
[21] Robert Elz and Randy Bush. 1997. Clarifications to the DNS Specification. RFC 2181. https://doi.org/10.17487/RFC2181
[22] Xun Fan, John Heidemann, and Ramesh Govindan. 2013. Evaluating Anycast in the Domain Name System. In Proceedings of INFOCOM '13.
[23] Google. 2023. Google Public DNS. https://developers.google.com/speed/public-dns/.
[24] P. Krishna Gummadi, Stefan Saroiu, and Steven D. Gribble. 2002. King: Estimating Latency between Arbitrary Internet End Hosts. In Proceedings of IMC '02.
[25] Paul E. Hoffman. 2023. DNS Security Extensions (DNSSEC). RFC 9364. https://doi.org/10.17487/RFC9364
[26] Paul E. Hoffman and Patrick McManus. 2018. DNS Queries over HTTPS (DoH). RFC 8484. https://doi.org/10.17487/RFC8484
[27] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. 2016. Specification for DNS over Transport Layer Security (TLS). RFC 7858. https://doi.org/10.17487/RFC7858
[28] Daniel Kopp, Christoph Dietzel, and Oliver Hohlfeld. 2021. DDoS Never Dies? An IXP Perspective on DDoS Amplification Attacks. In Proceedings of PAM '21.
[29] NLnet Labs. 2023. Unbound. https://www.nlnetlabs.nl/projects/unbound/about/
[30] Xiang Li, Baojun Liu, Xuesong Bai, Mingming Zhang, Qifan Zhang, Zhou Li, Haixin Duan, and Qi Li. 2023. Ghost Domain Reloaded: Vulnerable Links in Domain Name Delegation and Revocation. In Proceedings of NDSS '23.
[31] Xiang Li, Baojun Liu, Xiaofeng Zheng, Haixin Duan, Qi Li, and Youjun Huang. 2021. Fast IPv6 Network Periphery Discovery and Security Implications. In Proceedings of DSN '21.
[32] Xiang Li, Chaoyi Lu, Baojun Liu, Qifan Zhang, Zhou Li, Haixin Duan, and Qi Li. 2023. The Maginot Line: Attacking the Boundary of DNS Caching Protection. In Proceedings of USENIX Security '23.
[33] 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 IEEE S&P '24.
[34] Florian Maury. 2015. The iDNS Attack (Resolver Loop). https://indico.dns-oarc.net/event/21/contributions/301/
[35] maxmind. 2023. GeoLite2 Free Geolocation Data. https://dev.maxmind.com/geoip/geolite2-free-geolocation-data
[36] Mikrotik. 2023. About Mikrotik. https://mikrotik.com/aboutus
[37] Mikrotik. 2023. RouterOS FAQ. https://wiki.mikrotik.com/wiki/Manual:RouterOS_FAQ
[38] Soo-Jin Moon, Yucheng Yin, Rahul Anand Sharma, Yifei Yuan, Jonathan M. Spring, and Vyas Sekar. 2021. Accurately Measuring Global Risk of Amplification Attacks using AmpMap. In Proceedings of USENIX Security '21.
[39] Giovane C. M. Moura, Sebastian Castro, John S. Heidemann, and Wes Hardaker. 2021. TsuNAME: Exploiting Misconfiguration and Vulnerability to DDoS DNS. In Proceedings of IMC '21.
[40] Marcin Nawrocki, Mattijs Jonker, Thomas C. Schmidt, and Matthias Wählisch. 2021. The Far Side of DNS Amplification: Tracing the DDoS Attack Ecosystem from the Internet Core. In Proceedings of IMC '21.
[41] Yevheniya Nosyk, Maciej Korczynski, and Andrzej Duda. 2022. Routing Loops as Mega Amplifiers for DNS-Based DDoS Attacks. In Proceedings of PAM '22.
[42] PowerDNS. 2023. Change the way RD=0 forwarded queries are handled. https://github.com/PowerDNS/pdns/pull/12425
[43] PowerDNS. 2023. PowerDNS Recursor. https://doc.powerdns.com/recursor/index.html
[44] 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 IMC '20.
[45] Christian Rossow. 2014. Amplification Hell: Revisiting Network Protocols for DDoS Abuse.. In Proceedings of NDSS '14.
[46] Kyle Schomp, Tom Callahan, Michael Rabinovich, and Mark Allman. 2013. On measuring the client-side DNS infrastructure. In Proceedings of IMC '13.
[47] Ben Schoon. 2021. Google's 8.8.8.8 DNS Service Slows down Amid Massive Facebook Outage. https://9to5google.com/2021/10/04/google-dns-service-facebook-outage-slowdown/
[48] Raffaele Sommese, kc claffy, Roland van Rijswijk-Deij, Arnab Chattopadhyay, Alberto Dainotti, Anna Sperotto, and Mattijs Jonker. 2022. Investigating the Impact of DDoS Attacks on DNS Infrastructure. In Proceedings of IMC '22.
[49] J. Stewart. 2003. Dns Cache Poisoning - The Next Generation. (2003).
[50] Unbound. 2023. Fix not following cleared RD flags potentially enables amplification. https://github.com/NLnetLabs/unbound/commit/b12ab31ae36ae2b124748d37835d74dca15b161f
[51] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. 2014. DNSSEC and Its Potential for DDoS Attacks: A Comprehensive Measurement Study. In Proceedings of IMC '14.
[52] Chuhan Wang, YASUHIRO KURANAGA, Yihang Wang, Mingming Zhang, Linkai Zheng, lixiang, Jianjun Chen, Haixin Duan, Yanzhong Lin, and Qingfeng Pan. 2024. BreakSPF: How Shared Infrastructures Magnify SPF Vulnerabilities Across the Internet. In Proceedings of NDSS '24.
[53] Donghui Yang, Zhenyu Li, and Gareth Tyson. 2020. A Deep Dive into {DNS} Query Failures. In Proceedings of USENIX ATC '20.
[54] Ramin Yazdani, Roland van Rijswijk-Deij, Mattijs Jonker, and Anna Sperotto. 2022. A Matter of Degree: Characterizing the Amplification Power of Open DNS Resolvers. In Proceedings of PAM '22.
[55] 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 CCS '23.
[56] 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 IMC '23.
[57] Mingming Zhang, Xiang Li, Baojun Liu, Jianyu Lu, jianjun chen, Yiming Zhang, Xiaofeng Zheng, Haixin Duan, and Shuang Hao. 2023. DareShark: Detecting and Measuring Security Risks of Hosting-Based Dangling Domains. In Proceedings of SIGMETRICS '23.
[58] Yiming Zhang, Baojun Liu, Chaoyi Lu, Zhou Li, Haixin Duan, Jiachen Li, and Zaifeng Zhang. 2021. Rusted Anchors: A National Client-Side View of Hidden Root CAs in the Web PKI Ecosystem. In Proceedings of CCS '21.
[59] 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 Proceedings of USENIX Security '20.