Ghost Domain Reloaded: Vulnerable Links in Domain Name Delegation and Revocation

Xiang Li[*Φ], Baojun Liu[*Φ], Xuesong Bai[†], Mingming Zhang[*], Qifan Zhang[†]
Zhou Li[†], Haixin Duan[*‡§✉], and Qi Li[*§✉]
[*]Tsinghua University, [†]University of California, Irvine
[‡]QI-ANXIN Technology Research Institute, [§]Zhongguancun Laboratory
{x-l19, zmm18}@mails.tsinghua.edu.cn, {lbj, duanhx, qli01}@tsinghua.edu.cn
{xuesong.bai, qifan.zhang, zhou.li}@uci.edu

***Abstract*—In this paper, we propose PHOENIX DOMAIN, a general and novel attack that allows adversaries to maintain the revoked malicious domain continuously resolvable at scale, which enables an old, mitigated attack, Ghost Domain. PHOENIX DOMAIN has two variations and affects all mainstream DNS software and public DNS resolvers overall because it does not violate any DNS specifications and best security practices. The attack is made possible through systematically "*reverse engineer*" the cache operations of 8 DNS implementations, and new attack surfaces are revealed in the domain name delegation processes. We select 41 well-known public DNS resolvers and prove that all surveyed DNS services are vulnerable to PHOENIX DOMAIN, including Google Public DNS and Cloudflare DNS. Extensive measurement studies are performed with 210k stable and distributed DNS recursive resolvers, and results show that even after one month from domain name revocation and cache expiration, more than 25% of recursive resolvers can still resolve it. The proposed attack provides an opportunity for adversaries to evade the security practices of malicious domain take-down. We have reported discovered vulnerabilities to all affected vendors and suggested 6 types of mitigation approaches to them. Until now, 7 DNS software providers and 15 resolver vendors, including BIND, Unbound, Google, and Cloudflare, have confirmed the vulnerabilities, and some of them are implementing and publishing mitigation patches according to our suggestions. In addition, 9 CVE numbers have been assigned. The study calls for standardization to address the issue of how to revoke domain names securely and maintain cache consistency.**

I. INTRODUCTION

Domain names are often registered and abused by adversaries to conduct worldwide criminal activities, such as botnets [13], [56], phishing [103], [104], and spam or malware distribution [12], [127]. In practice, ICANN develops the Domain Abuse Activity Reporting system (DAAR) and publishes monthly reports of domain name threats. In April 2022, the system captured more than 633k newly-registered domains as malicious in just one month [64].

Nowadays, *domain name revocation* [111] is considered to be one of the best security practices and widely adopted to fight against malicious domain names employed by cyber-crime activities [38], [79]. Security operators can revoke a malicious domain name through forcibly deleting or changing delegation information from its parent zone with the help of registries or registrars [62]. After revocation, resolving the revoked domain (and its subdomains) should be either rejected, or redirected to nameservers owned by security operators.

Ensuring the effective domain name revocation is of great significance to the health of the DNS ecosystem, which mitigates various criminal activities. However, completely achieving this goal is not easy. This is because current DNS protocols [94], [95] do not require resolvers to *actively* fetch the up-to-date delegation information from the upper-level zones. Thus, domain name revocation has to take effect *passively* after the original cache expires in resolvers. Due to the complexity of the DNS cache mechanism, security risks may be introduced to prevent the effectiveness of domain name revocation.

Research gap. From an adversary's perspective, it is valuable to continually perform and control criminal activities (e.g., botnets, phishing, etc.), even after the malicious domain name has been revoked. In 2012, Jiang *et al.* [67] revealed a vulnerability in the DNS *cache update policy* and proposed an attack named *Ghost Domain*. It allows a malicious domain to be continuously resolvable at the global scale, even for one week after the domain was revoked. After disclosure of the ghost domain attack, all mainstream DNS software was patched at once [19], [39], [112], [138]. Ten years later, *it is commonly believed that the ghost domain attack has been eliminated*, and the desired outcome of domain name revocation should be guaranteed. However, we suspect that, as previous research primarily only focused on security threats of cache updating, the security community still lacks systematic analysis of all DNS cache operations, including cache searching and cache insertion. This research gap results in *the entire attack surface of domain name revocation remaining unclear now*.

Our study. To fill this research gap, in this paper, we select 8 mainstream DNS software and conduct a systematic "reverse engineer" study to understand their internal caching mechanisms. Through source code review and comprehensive black-box testing, surprisingly, we find that *loopholes of domain name revocation are not closed after ten years* and uncover general and novel attack surfaces that are hindering the security guarantees of domain name revocation. Similar to Ghost

[Φ]Both are first authors. ✉ Corresponding authors.

Network and Distributed Systems Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.23005
www.ndss-symposium.org

Domain, our proposed attack, PHOENIX DOMAIN, also allows adversaries to maintain a revoked domain staying resolvable over a very long time, overcoming current patches.

Currently, DNS RFCs do not explicitly state that domain name revocation should be satisfied as one of the basic domain properties, i.e., *consistency* that the delegation data from parent zones and child zones should be consistent (RFC 1034 [94]). Domain name revocation works by changing delegation data from parent zones, which should conform with the consistency property. In practice, DNS cache operations are all implemented from the perspectives of performance and efficiency. Our research demonstrates that a suit of cache properties can be exploited by adversaries to prevent resolvers from fetching the up-to-date delegation information (Section III).

Specifically, PHOENIX DOMAIN has two variations, named `T1` and `T2` (Section IV-B). $T1$ results from vulnerable DNS implementations while $T2$ stems from the generic DNS protocol specifications [94], [95]. (1) When managing cache, resolvers rely on the TTL-aging mechanism (time to live) to remove cached records after expiration. $T1$ exploits inconsistent *time-of-use* and *time-of-check* cache validation to refresh the delegation data via an attack time window, which is effective for specific DNS implementations. (2) During the local cache searching, resolvers prefer to trust the responses from the child zone than the parent zone and use *the closest known* nameserver for outgoing queries, following the de facto DNS specifications. $T2$ leverages this property to insert new child delegation data into the resolver under the cache miss state and use iterative subdomains to prolong the cache lifetime, thus applicable to all DNS implementations. Theoretically, attackers can keep (sub)domains in the cache for a 127× TTL time with $T2$ and potentially indefinitely with $T1$. *Since new delegation data is renewed with the former cached data missed or removed, we call the exploited domain Phoenix Domain.*

Key findings. PHOENIX DOMAIN breaks current mitigation patches against the original ghost domain attack. Our experiments show that all mainstream DNS software and 41 popular public DNS resolvers we surveyed are vulnerable to $T1$ and/or $T2$, in particular, BIND [21], Unbound [139], Google Public DNS [55], and Cloudflare DNS [30] (Section V-A and VI-C).

In addition, we performed extensive measurements to evaluate the real-world impact (Section VI) with practical attack considerations (Section V-B). First, we collected representative and stable open resolvers through a 2-month-long ongoing resolver scanning and carefully select *210k* recursive resolvers as our experiment targets with the consideration of ethics. Second, we created several domain names for comparison and performed both short-term and long-term experiments. Third, we also analyzed and tested different experiment settings that affected PHOENIX DOMAIN to show the attack effect and cost. Through experiments, we demonstrate that more than 89% of 210k recursive resolvers are vulnerable, and a large exploitation is practical. Results also show that more than 40% resolvers can successfully resolve the revoked domain name after one week for $T2$, while more than 25% one month later.

Disclosure and mitigation. We responsibly report the discovered vulnerabilities to all affected vendors. We also suggest 6 short-term and long-term approaches to mitigating PHOENIX DOMAIN attacks (Section VII). So far, 7 DNS software and 15 public resolver vendors have confirmed the vulnerabilities and are implementing mitigations according to our report. 9 CVE-ids have been assigned[1]. In the end, our study brings attention to re-examining the domain name delegation and revocation mechanism and calls for standardization and agreements to address PHOENIX DOMAIN and revoke domain names securely and maintain cache consistency.

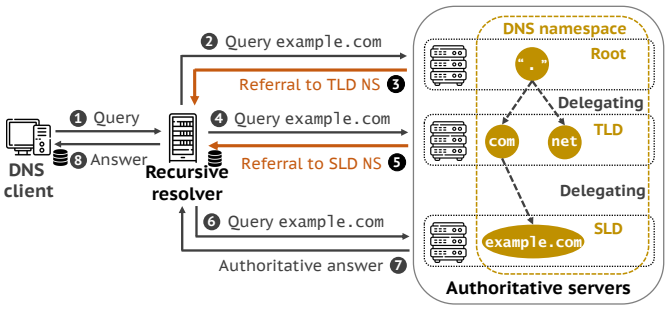


Fig. 1. DNS resolution process and DNS namespace.

Contributions. We make the following contributions:

Systematic analysis of implementations. We summarize potential DNS cache operation weaknesses through source code review and black-box experiments by analyzing 8 DNS software.

Novel attack. We propose PHOENIX DOMAIN to make a revoked domain name sticky to resolvers applicable to all DNS implementations via exploiting new attack surfaces introduced by $T1$ and/or $T2$.

Measurements to evaluate attacks. We conduct a month-long measurement to demonstrate the attack effect of 210k open recursive resolvers and test 41 public resolvers.

Disclosure and mitigation. We report vulnerabilities to all affected vendors and discuss 6 mitigation approaches to addressing PHOENIX DOMAIN attacks with them.

II. BACKGROUND

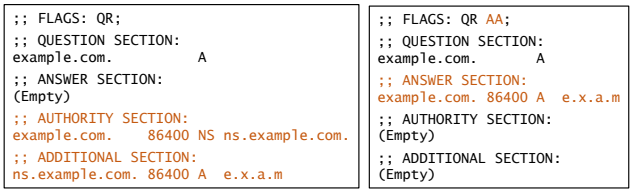
In this section, we overview the DNS resolution process and domain name delegation [94], [95]. Then, we explain DNS cache update policies. Finally, we describe domain name revocation process and Ghost Domain that breaks its guarantees.

A. DNS Overview

DNS namespace and resolution. DNS partitions the domain namespace into a hierarchical structure, with the higher and lower levels called parent and child. As shown in Figure 1 right, at the top of the hierarchy is the DNS *root* zone. Below the DNS root zone is a series of Top-Level Domains (TLDs), such as `.com`, `.net`, and `.org`. Under TLDs, Second-Level Domains (SLDs) are managed by registrars, such as GoDaddy [53] and Verisign [144], and open to registration. As an example, the TLD and SLD of `www.example.com` are `.com` and `example.com`.

Most DNS queries generated from clients are processed by recursive resolvers first. To resolve a DNS request, the

[1] PHOENIX DOMAIN: https://phoenixdomain.net/.



(a) Referral. (b) Answer.

Fig. 2. DNS referral and authoritative answer responses of `example.com`.

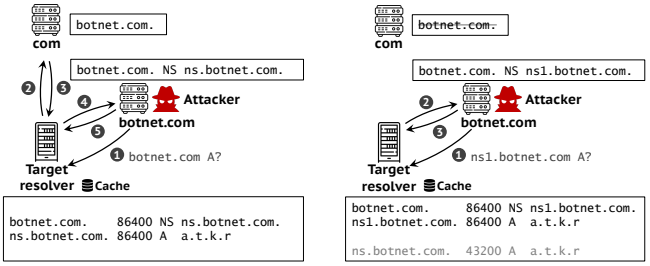
resolver will contact the DNS root server, TLD, and SLD nameserver subsequently, as shown in Figure 1 left (step ❷, ❹, and ❻). In this process, nameservers that are not authoritative for the requested domain respond with the *referral* information they have. For example, for the query about `www.example.com`, the `.com` nameserver returns the referral information (`example.com` nameserver) to the resolver (step ❸). Resolvers and clients are excepted to cache the received response data, including referrals and answers, to speed up subsequent queries about the same domain names.

DNS data structure. A DNS message (both request and response) contains four sections, namely "question", "answer", "authority", and "additional". Figure 2 shows two types of responses (signaled by `QR` in "FLAGS") when querying `example.com`. When the answer response comes from the authoritative nameserver (indicated by `AA` in "FLAGS"), the answer section is used to deliver the resource records stored in the DNS zone file of the requested domain (see Figure 2(b)). Otherwise (no `AA` in "FLAGS"), authority and additional sections are employed to give referral information (shown in Figure 2(a)).

A resource record embedded in a section is a 5-tuple: <*Domain Name, Time to Live, Class, Type, Value*>. Domain Name serves as a primary key in the DNS resolution. Type describes the kind of record, such as `A` for IPv4 addresses and `NS` for authoritative nameservers. Time to Live (TTL) determines the duration of a record to be cached. In Figure 2, the resource records of `example.com` should be cached for 86,400 seconds.

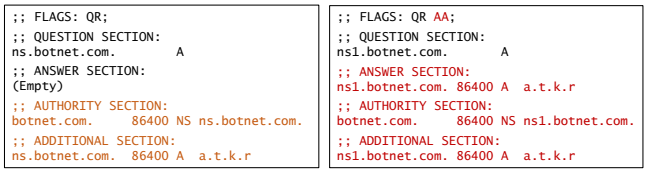
Domain name delegation. It occurs when a registrant obtains a domain name from a registry or registrar, or updates the zone files [63]. Since the domain namespace can be represented as a tree, each delegation is a tree branch, and the delegation data provide referrals for nameservers that are authoritative for the child zone. The delegation information can be either `NS` records (pointing to the child zone nameserver) or *glue records* (specifying the nameserver IP addresses). In a DNS response, typically, the authority section carries `NS` records, while the additional section carries glue records, as shown in Figure 2(a).

To make the recursive resolution possible, the parent zone must include delegation information for its child zones. For instance, the `.com` zone contain referrals pointing to nameservers that have authority for the `example.com` zone (step ❺ in Figure 1).



(a) Before `botnet.com` is revoked, the attacker makes the victim resolver cache delegation data of `botnet.com`. (b) After `botnet.com` is revoked, the attacker makes the victim resolver cache new delegation data of `botnet.com`.

Fig. 3. An example of the ghost domain attack. The gray text in box shows the records that become invalid.



(a) Normal referral response. (b) Attack answer response.

Fig. 4. DNS responses used by the ghost domain attack.

B. DNS Cache Update Policy

DNS caching decreases the query latency and reduces the traffic volume of authoritative nameservers [32], [49], [68], [97]. Any resolver or client can store the DNS responses and reuse them in the future until cache expiration. However, the situation becomes complicated when considering whether to *overwrite* resource records that are *already* in the cache, due to that the DNS data about a domain can come from different servers and sections. DNS RFCs define trust levels for DNS data [47], and the data with higher or equal ranking will be preferred to overwrite the existing one. The trust levels are generally defined under the two basic principles.

- **P1:** DNS answers from the authoritative server have a higher trust level than the non-authoritative answers.
- **P2:** Records from the answer section have a higher ranking than records from authority or additional sections within the same DNS response.

Despite the high-level policies, DNS RFCs did not offer concrete guidance for cache operations, leaving DNS software to implement them. Table I provides a detailed analysis of DNS software. As a highlight, we found inconsistency exists in preferring delegation records from child zones (*child-centric*) or parent zones (*parent-centric*). Noticeably, none of child-centric and parent-centric conflicts with *P1*. For child-centric, the delegation data received from the child zone is preferred according to *P1*. Whereas for parent-centric, resolvers ask parent zones for delegation data and ignore child zones (see Section V-A). Thus, *P1* does not apply since no data needs to be overwritten.

C. Domain Name Revocation and Ghost Domain Attack

Domain name revocation is the reverse process from domain name delegation, which cancels or changes the domain

ownership. When a registered domain expires without renewal, the registration will be canceled. If a domain name violates the policies defined by ICANN [61], [63], such as being used by botnet [13], domain sinkholing and delisting could be carried out [11], [62]. For domain sinkholing, the IP addresses or nameservers of the revoked domain will be changed to point to servers of the legal authorities. Domain delisting removes the domain from the domain namespace and responds with nonexistence (`NXDomain`) whenever there is a query about the domain.

For either of the above operations, the executors need to modify the delegation data in the parent zones from the revoked domain. Two guarantees are supposed to be achieved under revocation.

- **G1:** The revoked domain should not be resolved to its original destination by the resolvers after the TTL specified in the resource records expires.
- **G2:** The subdomains of the revoked domain should not be resolvable.

Ghost domain. In 2012, Jiang *et al.* [67] showed that an attacker could break the guarantees with the ghost domain attack, and many mainstream DNS software were affected, including BIND and PowerDNS. Specifically, the attacker can keep a domain resolvable at resolvers even after revocation. The attack takes multiple steps, and we follow the example shown in Figure 3 and Figure 4. First, the attacker actively asks a target resolver to resolve the malicious domain `botnet.com` to keep the original delegation data being cached (Figure 3(a) and Figure 4(a)). Second, after revocation (`botnet.com` and `ns.botnet.com` are removed from `.com` zone), the attacker changes the `NS` records of the malicious domain to new records (`ns1.botnet.com`) at his/her authoritative nameservers. At last, before the TTL of the cached delegation data expires in the resolver, the attacker queries a subdomain of `botnet.com` that can be processed by his/her authoritative nameservers (Figure 3(b)). As a result, the delegation information about `botnet.com` is refreshed by the attacker illegally, including a new TTL (86,400 seconds), which breaks guarantee *G1* (`botnet.com` is still resolvable) and *G2* (the nameservers of `botnet.com` are cached, which can be queried to resolve `botnet.com`'s subdomains).

Mitigation. Ghost domain exploits the vagueness of the cache update policy. *P1* specifies that the answers from the authoritative nameservers are more trustworthy, but it should not be the case when the associated domains are revoked. Under the ghost domain, the cached delegation data (`NS` records) can be overwritten by the responses from the nameservers of the revoked domain (policy *P1*). After its disclosure, mainstream DNS software have corrected their DNS cache policies [19], [39], [112], [138]. Currently, resolvers still allow authoritative nameservers to update their delegation information, *except the TTL value* of the authoritative data [121], [137], thus invaliding the ghost domain attack. Following the above example, the TTL of `botnet.com`'s `NS` records will not be restored to 86,400 when the resolver receives the response in Figure 2(b).

III. Systematic Analysis of DNS Cache

Though the majority of DNS software has been patched against the ghost domain attack (shown in Table I), we suspect that the threat is not fully mitigated due to the complexity of DNS cache and the inconsistency among different software. In this section, we systematically "*reverse engineer*" the cache mechanisms of the mainstream DNS software through code review and software testing to analyze the internal DNS cache operations and uncover new attack surfaces.

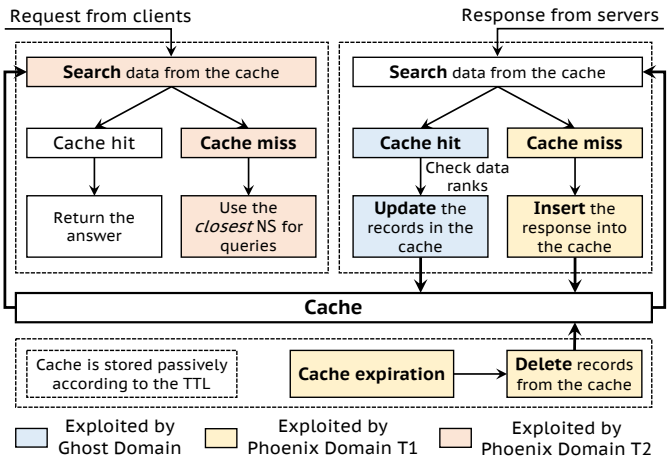


Fig. 5. DNS cache operations by recursive resolvers.

A. Generic Workflow of DNS Cache

To gain deeper understanding of the workflow of DNS cache, we analyzed the latest version of 8 popular DNS software that support the recursive resolver mode, including BIND9 [21], Knot Resolver [74], Unbound [139], PowerDNS Recursor [114], Microsoft DNS [93], Simple DNS Plus [128], Technitium [135], and MaraDNS (Deadwood) [87], as shown in Table VII. The above DNS software are also selected by previous works that discovered prominent DNS vulnerabilities [4], [66], [67], [71], [77], [81], [84], [85], [147]. Each software has tens or hundreds of thousands of SLOC (Source Lines of Code), and it took about two weeks to analyze them all together, as summarized in Appendix A.

By reviewing the source code of DNS software (except Microsoft DNS and Simple DNS Plus), we found that DNS cache operations are tightly integrated into the resolution process, and they are *passively* triggered along with a DNS request or response. Figure 5 shows the generic workflow. Upon receiving a DNS request, the resolver extracts the primary search key (<*Domain Name, Class, Type*>) and searches the local cache database. If the search key has a hit, the resolver returns the answer directly to the client. Otherwise, the resolver searches for an authoritative nameserver and contacts it.

Upon receiving a DNS response from nameservers, the resolver examines whether the received resource records exist in the local cache. If not, the records will be inserted into the cache database. Otherwise, the resolver applies the trustworthiness rules (described in Section II-B) to determine whether to overwrite the cache.

Independent from DNS requests and responses, the deletion of DNS cache entries is performed passively and triggered when the record TTLs expire.

B. Attack Surface

Though the ghost domain attack can evade domain revocation, we found it only considers updating a cache entry, leaving out the other cache operations, i.e., cache insertion and cache searching. Below we analyze these two cache operations and show their potential weaknesses. In Section IV-B, we show concrete exploitation.

DNS cache insertion. After receiving a DNS response from the nameserver, resolvers are excepted to cache the resource records that are embedded in answers and referrals. As presented in Figure 5, the execution flows differ based on cache hit and cache miss. We found though stricter policies have been applied in cache update following cache hit due to the disclosure of Ghost Domain (e.g., no extension of TTL, described in "Mitigation" of Section II-C), there is no restriction on cache insertion after cache miss, leading to new attack surfaces.

Specifically, because cache deletion is triggered by TTL expiration and the process is *detached* from DNS resolution, there exists a time window for the attacker to launch a *time-of-check and time-of-use (TOCTOU)* attack. The attacker can query a subdomain (assuming the subdomain has not been cached) to let the resolver contact his/her nameserver, delay the response till the original delegation data is removed, and reply with new delegation data. This time, cache insertion is triggered following cache miss (due to the removal of old delegation data), and the security policies deployed around cache update are all bypassed.

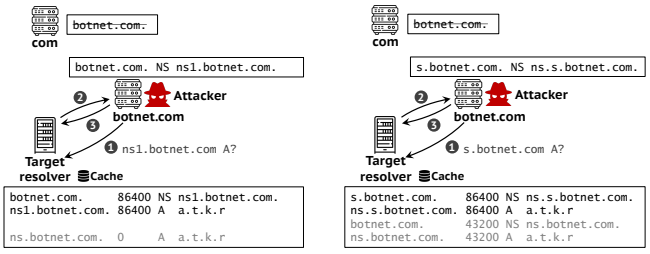
DNS cache searching. After receiving a DNS request, the resolver examines whether the queried record is cached. When cache miss happens, the resolver needs to contact an authoritative nameserver. It turns out DNS software perform fuzzy matching to locate *the closest known delegation information* and choose the authoritative nameserver from it, following DNS RFC (Section 5.3.3 in RFC 1034 [94]). This is because the delegation records from the child zones are considered more trustworthy than records from the parent zone [47], [131], following policy *P1* described in Section II-B.

As an example, we provide the pseudo-code of BIND9 cache searching in Algorithm 1 (in Appendix B). When there is a cache miss, fuzzy matching is done through the *Longest Suffix Match* (*LSM*) algorithm (Line 12 of Algorithm 1). LSM algorithm starts by searching nameservers of the queried domain name, then the parent domain name, the grandparent, and so on toward the root. As regards to the example of Figure 1, if the referral response in Figure 2(a) has been cached, the resolver directly contacts nameservers of `ns.example.com`.

Though fuzzy matching reduces unnecessary queries to upper-level zones, it prevents resolvers from getting timely and up-to-date delegation information from parent zones. Hence, the attacker can exploit the child zone to provide answers different from the parent zones and evade revocation.

IV. Attack Overview

In this section, we describe our attack Phoenix Domain that exploits the attack surfaces from cache insertion and searching, and breaks the guarantees of domain name revocation. Phoenix Domain does not violate any DNS RFCs and affects mainstream DNS software and popular public DNS servers. First, we describe the threat model of our attack. Then, we dive into details of the two exploitation, $T1$ and $T2$, and compare them with the original ghost domain attack.



(a) After `botnet.com` is revoked, the attacker makes the victim resolver cache new delegation data of `botnet.com` when `ns.botnet.com` is about to expire ($T1$).

(b) After `botnet.com` is revoked, the attacker manipulates the victim resolver to cache delegation data of `s.botnet.com` before `ns.botnet.com` expires ($T2$).

Fig. 6. Examples of the Phoenix Domain attack. The gray text in box shows the records that become invalid.

A. Threat Model

Like the ghost domain attack, we consider an adversary who aims to keep the malicious domain alive after revocation by "poisoning" the cache of recursive resolvers. We assume the attacker controls the authoritative nameservers of the revoked domain, and the control persists after revocation (e.g., attackers can set up their nameservers with bullet-proof hosting, which is resilient against legal actions [11], [62]). The attacker is able to query the target resolvers with arbitrary DNS requests. When the target resolvers are open, e.g., Google Public DNS [55], this condition is automatically satisfied, and Section VI shows that numerous open resolvers can be attacked. When the target resolvers are private, the attackers need to join their networks or hijack a victim client in the same network (e.g., by tricking the client into visiting a malicious web page and executing DNS queries). We also assume that the attacker knows when the domain is about to be revoked (e.g., getting notified from the registrars), so the attack can be performed right away. Though domain revocation assumes a "passive defender" who waits for the cached records of the malicious domain on the resolvers to expire, our attack is also resilient against an "active defender" who tries to query the target resolvers to force cache updates, because the resolvers have to inquire the attackers' nameservers subsequently.

B. Attack Workflow

Here we use the same example of the ghost domain attack (Section II-C) to demonstrate the workflow of $T1$ and $T2$. For both variations, in the first stage, the attacker requests the target resolver to resolve `botnet.com` (Figure 3(a)), and the nameservers of `.com` return delegation data of `botnet.com` to the resolver, and the resolver accepts and caches the data for a TTL time, 86,400 seconds. Then, `botnet.com` is revoked from the `.com` zone, while the target resolver still caches its `NS` records, `ns.botnet.com`. In the second stage, before cache expiration, the attacker can conduct $T1$ and $T2$ respectively to keep delegation records of `botnet.com` or its subdomains continuously alive. Starting from here, we use Figure 6 to show the exploitation process.

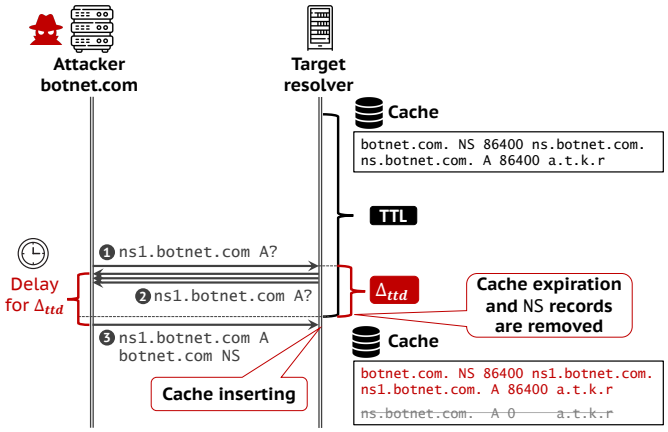


Fig. 7. PHOENIX DOMAIN $T1$ attack steps.

```
;; FLAGS: QR AA;
;; QUESTION SECTION:
ns1.botnet.com.        A
;; ANSWER SECTION:
ns1.botnet.com. 86400 A  a.t.k.r
;; AUTHORITY SECTION:
botnet.com.     86400 NS ns1.botnet.com.
;; ADDITIONAL SECTION:
ns1.botnet.com. 86400 A  a.t.k.r
```

(a) Answer for $T1$.

```
;; FLAGS: QR;
;; QUESTION SECTION:
ns1.botnet.com.        A
;; ANSWER SECTION:
(Empty)
;; AUTHORITY SECTION:
botnet.com.     86400 NS ns1.botnet.com.
;; ADDITIONAL SECTION:
ns1.botnet.com. 86400 A  a.t.k.r
```

(b) Referral for $T1$.

```
;; FLAGS: QR AA;
;; QUESTION SECTION:
s.botnet.com.          A
;; ANSWER SECTION:
s.botnet.com.   86400 A  a.t.k.r
;; AUTHORITY SECTION:
s.botnet.com.   86400 NS s.botnet.com.
;; ADDITIONAL SECTION:
s.botnet.com.   86400 A  a.t.k.r
```

(c) Answer for $T2$.

```
;; FLAGS: QR;
;; QUESTION SECTION:
s.botnet.com.          A
;; ANSWER SECTION:
(Empty)
;; AUTHORITY SECTION:
s.botnet.com.   86400 NS s.botnet.com.
;; ADDITIONAL SECTION:
s.botnet.com.   86400 A  a.t.k.r
```

(d) Referral for $T2$.

Fig. 8. DNS payloads utilized for $T1$ and $T2$ on `botnet.com`.

***T1* attack.** As illustrated in Figure 6(a), the attacker manipulates the target resolver to accept and cache the new delegation data of `botnet.com` when the old cached data *is about to expire*. We exploit a delicate attack time window to achieve the $T1$ attack, as shown in Figure 7.

After $86,400 - \Delta_{ttd}$ (the time to the TTL expiration) seconds from the initial caching of `botnet.com`, the delegation data is about to expire within Δ_{ttd} seconds. At this moment, because the cached data is still valid, attackers can make the resolver query a subdomain that has not been encountered (e.g., `ns1.botnet.com`) towards the nameserver of `botnet.com` (steps ❶ to ❷). On the nameserver, after receiving the query, the attacker delays the response for Δ_{ttd} seconds (step ❸). Therefore, when the resolver receives new delegation records carried by either the answer or referral response[2] in Figure 8(a) or Figure 8(b), due to passive cache management and inconsistent cache use-and-check operations shown in Section III-A, old delegation data has expired and been removed from the cache (although just been used), and new data will be stored for *a refreshing TTL* of 86,400 seconds without violating current cache update policy. In the following exploiting round, attackers can conduct the same delay-and-respond operation again and again when the cached delegation data is about to expire, which makes `botnet.com` and all subdomains resolvable for a long time (*potentially indefinitely*).

In the end, $T1$ bypasses all patches related to the ghost domain attack and breaks guarantee *G1* and *G2* in Section II-C. A concrete problem the attacker needs to address is how to delay the response without aborting the resolution process, as the longer the delay, the more likely $T1$ will succeed. We discuss how to extend the time window in Section V-B.

***T2* attack.** Before `NS` records of `botnet.com` expire, the attacker can request the target resolver to query a subdomain of `botnet.com` (`s.botnet.com`) that has not been encountered. Because of cache miss, the resolver searches for the closest `NS` record in cache and contact the nameserver. As shown in Figure 6(b), after 43,200 seconds, the cached `NS` record of `botnet.com` is still valid with a remaining lifetime of 43,200 seconds. On the authoritative server, the attacker can create *a new downward delegation* and return new delegation data of `s.botnet.com` to the resolver. The delegation payload can also be carried by answer or referral responses[2] in Figure 8(c) and Figure 8(d). Then, the target resolver accepts and caches the delegation response for *a new TTL* of 86,400 seconds. Attackers can use subdomains of `s.botnet.com` for communication. After another 43,200 seconds, the delegation data of `botnet.com` expires, but the data of `s.botnet.com` is still alive.

As mentioned in Section III-B, due to the use of fuzzy matching in cache searching, queries for subdomains under `s.botnet.com` are still sent to the nameservers of `s.botnet.com` by the resolver. Therefore, the attacker can delegate another subdomain under `s.botnet.com`, such as `s.s.botnet.com`, and repeat this process by adding new prefix labels. Given that the number of the labels (or the recursive levels) of a domain name is up to 127, theoretically, the attacker is able to keep subdomains in the resolver's cache for a time of *127× TTL* with the above strategy. In Section V, we show the common maximum TTL value is 7 days, so the cache lifetime can be more than *2 years*.

In the end, $T2$ enables the resolution of subdomains of a revoked domain, which breaks guarantee *G2* in Section II-C.

Remarks. In this section, we describe the technical details of two exploitation, $T1$ and $T2$. A classical usage scenario of $T1$ and $T2$ attacks is botnet C&C communication. Under $T1$, the attacker can keep `botnet.com` as the contact point (e.g., C&C domain) to the cyber-crime victims for a very long time. Under $T2$, attackers can also leverage a fixed domain name (*the longest domain name* with the most labels), e.g., a subdomain of `botnet.com` with 122 "`s`" prefix labels, as the contact point from the exploitation beginning.

In addition, $T1$ and $T2$ can apply to the phishing attack. In the previous study, Du *et al.* described an emerging new type of domain-name fraud, named *level-squatting* [45]. Unlike existing frauds that use similar second-level domain names to impersonate users, adversaries embed brand names in the subdomain section to deceive users, especially mobile users. Since some mobile browsers fail to display level-squatting domains, Internet users may be vulnerable to this fraud. For example, attackers could use `paypal.com.botnet.com` under $T1$ while adopting `paypal.com.s.`$\cdots$`.s.botnet.com` with 114 "`s`" labels in the middle for $T2$.

[2] The delegation data in both the answer and referral response are cachable. But for the referral, the recursive needs to query again for a final answer.

TABLE I. THE RESOLUTION MECHANISM IMPLEMENTATION DIFFERENCES OF MAINSTREAM DNS SOFTWARE FOR RECURSIVES.

Software	Version	Answer data[1]	Referral data[1]	Delegation update policy	Max TTL	Max timeout	Query count	DNSSEC	Vulnerable?				
									G_O	$T1_A$	$T1_R$	$T2_A$	$T2_R$
BIND9 [21]	9.18.6	✓	✓	Child-centric	7 days	10s	13	✓	✗	✓[2]	✓	✓[2]	✓
Knot Resolver [74]	5.5.2	✓	✓	Child-centric	6 days	1.2s	3	✓	✓[3]	✓	✓	✓	✓
Unbound [139]	1.16.1	✓	✓	Child-centric	1 day	>10s	9	✓	✗	✓	✓	✓	✓
PowerDNS Recursor [114]	4.7.2	✓	✓	Child-centric	1 day	1.5s	1	✓	✗	✗	✗	✓	✓
Microsoft DNS [93]	2022	✓	✓	Parent-centric	1 day	3s	1	✓	✗	✗	✗	✓	✓[4]
Simple DNS Plus [128]	9.1.108	✓	✓	Parent-centric	7 days	1.5s	3	✓	✗	✗	✗	✓	✓[4]
Technitium [135]	8.0.2	✗	✓	Parent-centric	7 days	10s	6	✓	✗	✗	✓[5]	✗	✓
MaraDNS [87]	3.5.0021	✗	✓	Parent-centric	1 day	6s	6	✗	✗	✗	✗	✗	✓[4]

[1] Whether caching delegation data of the authoritative section (`NS` records) and additional (glue records) section from answer or referral responses.
[2] Not vulnerable if the DNSSEC option is turned on by default.
[3] Vulnerable if the remaining TTL value is less than 5s or 1% of original TTL.
[4] Vulnerable if delegating to a new IP address.
[5] Vulnerable if delegating to a new `NS` name different from the queried domain name.
✓: Yes. ✗: No. ✓: Vulnerable. ✗: Not vulnerable.

Discussion. In general, $T2$ can also work under a domain generation algorithm (DGA). We compare the $T2$ DGA with traditional DGA here. To avoid the disruption from domain revocation, the traditional DGA would require the attacker to register many domains from registrars or free dynamic DNS (DDNS) services (e.g., No-IP [101]). For the first case, recently, some registrars and authorities put stricter policies on bulk domain registration with APIs [29], [90]. For the second, that a domain is registered under DDNS services has been considered as a detection feature [91], [110].

$T2$ overcomes the above limitations by just using a revoked domain and does not require registering new domains. The major issue with $T2$ is that when the attack runs for long, the malicious subdomain will have many labels, distinguishing itself from the normal domains. This is due to the $T2$ domain should be changed after each attacking round (e.g., from `botnet.com` to `s.botnet.com` to `s.s.botnet.com`). Therefore, a DGA needs to be run on the victim side to update the contact point. We will discuss this issue in Section VII-A.

C. Comparison to Ghost Domain Attack

Similar to Ghost Domain, PHOENIX DOMAIN aims to keep malicious domains alive in the target resolvers after domain name revocation. However, PHOENIX DOMAIN differs from the ghost domain attack by uncovering *a broader attack surface* related to DNS cache operations and exposing security threats to domain name revocation mechanisms even after the patches against the ghost domain attack have been widely adopted.

In particular, the ghost domain attack exploits the vulnerabilities in cache update implementations to prolong the lifetime of cached delegation records. As a result, these vulnerabilities are specific to DNS software that adopts a problematic child-centric delegation update policy without *restricting the TTL value refreshing*. In contrast, $T1$ exploits the vulnerability of inconsistent cache use-and-check operations when inserting data in cache expiration and affects specific DNS implementations, as shown in Table I. However, $T2$ exploits de facto cache searching operations to insert data in cache miss, stemming from generic DNS protocol standards and applicable to all DNS implementations.

V. FEASIBILITY ANALYSIS OF PHOENIX DOMAIN

To demonstrate the feasibility of PHOENIX DOMAIN attacks, we provide the evaluation results of the 8 analyzed DNS software in this section. We also describe the practical considerations in launching PHOENIX DOMAIN under real-world constraints.

A. Vulnerable DNS Software

As described in Section IV, we surveyed 8 mainstream DNS implementations. Through source code review and black-box experiments, we analyze and test their caching behaviors, including the cache update policy and maximum TTL value in default, which are illustrated in Table I. Based on these results, we are allowed to validate whether they are vulnerable to PHOENIX DOMAIN and the original ghost domain attack. For convenience, we abbreviate the original ghost domain attack as G_O. Besides, in Figure 8, we showed that the attacker's answers could be embedded in different responses. Therefore, we further name the $T1$ with the *answer* response (*referral* response) as $T1_A$ ($T1_R$), same to $T2$ as $T2_A$ ($T2_R$).

Vulnerable implementations. First, we inspect the cache accepting behaviors, i.e., which section from what DNS response is cachable. As shown in the column of "Answer data" and "Referral data", all software except Technitium and MaraDNS will cache the delegation data from both the answer and the referral response. Thus, Technitium and MaraDNS should be immune to $T1_A$ and $T2_A$ when the answer response is utilized for attack (Figure 8(a) and Figure 8(c)).

Second, when updating cached delegation data, Microsoft DNS, Simple DNS Plus, Technitium, and MaraDNS prefer to trust data in the referral response from the parent zone (*parent-centric*). Specifically, for Microsoft DNS, the TTL field of cached `NS` records can be updated while the name and IP address of authoritative servers cannot be changed. This behavior is introduced by the DNS cache locking security feature [92] that the resolver will not overwrite cached entries

for the entire duration of the TTL by default. Simple DNS Plus overwrites cached glue records but does nothing to cached NS records encountering new delegation data. Technitium just ignores unsolicited records in the authority and additional section and uses cached delegation data from parent zones as an answer to NS queries. MaraDNS takes a distinct way to cache delegation data that can only be fetched from the parent zone (just caching the domain name and nameserver's address as a key-value pair [132]). Therefore, they are not affected by G_O, $T1_A$, and $T1_R$, and we summarize them as adopting parent-centric update policies. The other 4 software choose delegation data from child zones prior to parent zones (*child-centric*) and are vulnerable to $T1_A$ and $T1_R$.

In summary, we find *all DNS software are vulnerable under* $T2$. *BIND9, Knot Resolver, Unbound, and Technitium are vulnerable under* $T1$. A caveat exists for BIND9, which ignores the delegation data in the authority and additional section from the answer response when the DNSSEC option is enabled, thus immune to $T1_A$ and $T2_A$. Surprisingly, PowerDNS Recursor adopts consistent cache use-and-check operations and is not affected by $T1$ (discussed in Section VII-A). As an exception, when the cached data expires, Technitium accepts referral responses from the child zone and is *susceptible* by $T1_R$. However, the nameserver's name should differ from the queried domain name. Besides, to exploit Microsoft DNS, Simple DNS Plus, and MaraDNS with $T2_R$, attackers should delegate nameservers to new IP addresses. We also obverse that Knot Resolver is still exposed to the risk of G_O until now. When the remaining TTL value of cached data is *less than 5s or 1% of original TTL*, the original ghost domain attack can still work against Knot Resolver.

B. Practical Attack Considerations

In this part, we discuss several practical attack considerations that affect the lifetime of PHOENIX DOMAIN.

Maximum cache TTL. Theoretically, an attacker can keep the malicious DNS answer in the cache for a very long time by setting a tremendous TTL value ($2^{31}-1$ seconds, around 68 years). However, previous research has shown that most recursive resolvers may not strictly comply with the TTL mechanism [83], [96], [123], and DNS operators might adopt a maximum TTL limitation [22], [73], [113], [141]. As a result, resolvers can overwrite the original TTL value when it exceeds the maximum value allowed. $T2$ has a total cache lifetime up to 127× TTL. For example, $T2$ can live for more than 2 years (with a 7-day TTL) and more than 4 months (with a 1-day TTL). We also re-examined the maximum TTL limitation for mainstream DNS implementations. As shown in Table I, all popular software have a maximum cache TTL of more than 1 day, and 3 of them have a maximum 7-day TTL by default (BIND9 [21], Simple DNS Plus [128], and Technitium [135]).

Maximum query timeout (attack time window). The maximum query timeout is the amount of time that resolvers spend attempting to resolve a recursive query before a failure. During the timeout window, resolvers will *retry* its current resolution with more than one outgoing queries (listed in the "Query count" column). The bigger the query timeout window is, the more success probability $T1$ will have. With *the transmission mechanism*, we can extend the attack time window to the maximum query timeout of each resolver. As shown in Table I, all implementations affected by $T1$ have a maximum query timeout of more than 10 seconds except for Knot Resolver (1.2 seconds). Through repetitious tests using BIND9 in our experimental network, we achieve a success rate of 100% out of 100 experiments with a ≥ 3 seconds timeout.

Multiple frontend caches. Many large or public DNS resolvers often have multiple frontend servers or multi-layer distributed caches that serve different queries even from the same client IP address with load-balancing or IP anycast [8], [52], [72], [118]. Therefore, to cover as many as caches, in each attack round, attackers can send multiple queries to each resolver. We will evaluate this method in Section VI.

Cache losses. Although DNS cache should be stored for the full TTL, resolvers may occasionally lose cached data due to cache capacity limit and cache flushing. Caches are of limited size. For example, the default size limit of BIND9 is 90% of physical memory [22]. When the amount of cached data reaches the limit, resolvers start evicting non-expired records based on cache replacement policies [32], e.g., BIND9's LRU-based strategy (Least Recently Used). For this case, frequent probing can help increase the cache hit rate, and previous works like [123] show that cache evictions due to capacity limits occur infrequently. Besides, the cache can be flushed explicitly by the resolver or accidentally due to software or machine reboot. In practice, this may reduce the cache hit rate, thus alleviating the impact of PHOENIX DOMAIN.

VI. FINDING VULNERABLE RESOLVERS IN THE WILD

In this section, by performing extensive experiments and measurements, we evaluate the real-world impact of PHOENIX DOMAIN with realistic settings. Note that, for $T1$, we only examine whether well-known public DNS services are exploitable without large-scale measurements. This is because to launch $T1$, the attack time window is required to be adjusted delicately for each DNS resolver. However, achieving time synchronization between all distributed devices remains a challenge [107]. Then, we provide our considerations of the potential ethical issues.

A. Collecting DNS Resolvers

Public resolver list. We collect a comprehensive resolver list of 41 popular public and free recursive vendors through search engines listed in Table VIII (in Appendix C) that take a top uses share for world [14], part of which are used in prior works as well [4], [66], [67], [71], [72], [84], [85], [118], such as Google Public DNS [55] and Cloudflare DNS [30]. We choose the primary IP address of each resolver for testing.

Open and stable resolver list. To collect open DNS resolvers, we take a straightforward approach by querying domain names, whose name servers are controlled by ourselves, toward UDP port 53 in the IPv4 address space using XMap [80]. We can observe open resolvers' ingress and egress IP addresses from both the client and name server sides. We regard the IP addresses that return correct DNS answers as potential open resolvers, of which the detailed statistics are listed in Table II.

We acknowledge that scanning from a single vantage point can not collect a comprehensive and complete resolver list.

TABLE II. STATISTICS OF ANALYZED OPEN RESOLVERS.

Resolver type	# IP	%
Open resolver	**1,202,041**	**100.0%**
Responsive on 02/18/2022	1,499,110	-
Responsive on 03/28/2022	1,202,041	80.2%
With valid PTR records	475,909	39.6%
With same AS for ingress & egress IP	991,155	82.5%
Recursive resolver	**210,029**	**17.5%**
Identified by ingress & egress IP	42,156	20.1%
Identified by AS of ingress & egress IP	1,689	0.8%
Identified by PTR records	12,210	5.8%
Identified by AS info & PTR records	153,974	73.3%

However, scanning the IPv4 network for DNS resolvers and other open services is widely used in security research, such as [65], [86], [100], [109]. We believe the discovered resolvers reflect the lower bound for us to evaluate PHOENIX DOMAIN.

During DNS probings by February 18, 2022, we obtained *1,499,110* possible resolvers, while only *1,202,041* (80.2%) remained alive over a month later (on March 28, 2022). To evaluate the validity of the remained resolvers, we further perform the following checks (adopted by [109] as well): First, we find that only *475,909* (39.6%) of these IP addresses have valid PTR records. Second, we compare the autonomous system (AS) information of resolvers' ingress and egress IP pairs and find *991,155* (85.2%) pairs belong to the same ASes.

To avoid potential ethical issues, we filter out forwarders since they tend to be the residential home routers [147] that do not perform as recursives (same ingress and egress IPs or AS information) and are neither equipped with PTR records nor DNS keywords [109]. Overall, we have selected *210,029* IP addresses that behave as recursives, representing 17.5% of all open resolvers. The following heuristic methods complete the selection: First, we treat the IPs as recursives if their ingress and egress IPs are the same. We obtain *42,156* (20.1%) recursives through this condition. Second, we leverage the AS information to indicate a candidate recursive, such as the organization name (e.g., AS 51289 of SkyDNS [129]), and acquire *1,689* (0.8%) recursives. Third, a feasible way is to identify keywords (e.g., "resolver" and "dns") in the PTR data of IP addresses, and we have selected *12,210* (5.8%) recursive resolvers. At last, for the left IPs, we combine the AS information and PTR data (same ASes of ingress and egress IPs, and valid PTR records) to check whether they are recursives implicitly. Results show that we can obtain the most (*153,974*, 73.3%) recursives. Furthermore, to evaluate the accuracy of the methods, we apply them to the public resolvers listed in Table VIII and identify *38* out of 41.

Distribution of recursive resolvers. We analyzed the geography and AS information of all recursive resolvers using the GeoLite2 database [89] and listed the top ten regions and ASNs in Table III. Results show all recursives' IP addresses belong to *218* regions, among which the top three are USA, China, and Russia. Looking at AS information, we find that IP addresses are under the control of *11,274* ASes. Of note, the distribution of global recursive resolvers may introduce the bias that a few regions account for the majority of resolvers. However, we aim to demonstrate the prevalence of affected resolvers worldwide rather than showing their geographical distribution in this study.

TABLE III. REGION AND AS DISTRIBUTION OF RECURSIVE RESOLVERS USED IN OUR EXPERIMENTS.

Region	Number	%	ASN	Number	%
USA	43,034	20.5%	4837	9,825	4.7%
China	25,152	12.0%	4134	5,988	2.9%
Russia	22,802	10.9%	3462	5,864	2.8%
Japan	13,421	6.4%	4713	5,134	2.4%
France	12,801	6.1%	8866	4,884	2.3%
Turkey	8,389	4.0%	9121	4,779	2.3%
Brazil	7,128	3.4%	16276	4,355	2.1%
Sweden	7,026	3.3%	209	3,937	1.9%
Taiwan	6,869	3.3%	3215	3,735	1.8%
Ukraine	6,572	3.1%	12389	3,485	1.7%
Total 218 regions			**Total 11,274 ASes**		

TABLE IV. SOFTWARE VERSION OF RECURSIVE RESOLVERS USED IN OUR EXPERIMENTS.

Software	# Resolver	Software	# Resolver
Microsoft DNS	38,484 (18.3%)	BIND	37,766 (18.0%)
PowerDNS Recursor	7,958 (3.8%)	Unbound	7,348 (3.5%)
Akamai	6,292 (3.0%)	Cleanbrowsing	2,194 (1.0%)
Simple DNS Plus	218 (0.1%)	Knot Resolver	8
Total identified	126,382 (60.2%)	Others	26,114 (12.4%)

Software version of recursive resolvers. For each recursive resolver, we adopt `version.bind` DNS queries [20] and an open-source DNS tool `fpdns` [42] to fingerprint their software versions. Overall, due to the resolver configuration and limited fingerprints, we only identified software versions of 60.2% out of 210k resolvers. We present the result details in Table IV.

B. Measurement Setups

To discover vulnerable resolvers in the wild, we conduct several measurements on both 41 public resolvers and 210k open resolvers. For each measurement, we use all resolvers to periodically trigger specially crafted DNS requests with dedicated *refreshing intervals* and *TTLs* toward domain names managed by ourselves. Then we probe and check whether we can receive valid responses from the client-side, determining the cache status of each domain name. In more detail, we create 5 groups of domain names (subzones[3]) under `botnet.com`[4] for comparison.

D1: Legitimate domain. We first conduct one controlled experiment by querying a live legitimate domain name, e.g., `google.com`, to check whether resolvers are stably alive.

D2: Revoked domain. We set another controlled experiment with a revoked domain name, `revoked.botnet.com`. We use it to show the cache status after the domain revocation.

D3: Ghost domain. To check whether the ghost domain is still exploitable in the real world, we reproduce experiments in [67]. For each refreshing round, we query a new domain name in the format of `ns<id>.ghost.botnet.com` with a different `id` plus one before the cache expires.

D4: Phoenix domain $T1$. This domain is also in the format of `ns<id>.phoenix1.botnet.com`, of which the `id` is plus one each refreshing round. When the cache is about to

[3] The revocation of subdomains works the same as the TLD, so we use subdomains for measurements instead of several TLDs.

[4] We registered a domain by ourselves for testing and use `botnet.com` as an example here for anonymity.

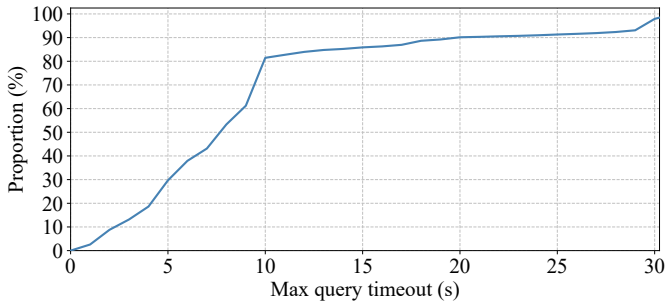


Fig. 9. The fraction of max query timeout of analyzed recursive resolvers.

expire (e.g., the remaining cache lifetime is 2s), we query a new subdomain and reply after a delay of Δ_{ttd} (e.g., 5s).

D5: Phoenix domain $T2$. We query the subdomain under `phoenix2.botnet.com` with a newly added sub-label prefix before cache expiration. For instance, after two refreshing intervals, we query `s.s.s.phoenix2.botnet.com` containing 3 "`s`" labels.

Notice that all domain names mentioned above and their authoritative servers are controlled by ourselves to set TTLs, answer data, and reduce the real-world impact. After the first testing round, we remove `NS` records of *D2-D5* from the `botnet.com` zone (conducting domain name revocation). Besides, for convenience, we leverage the *answer* responses in Figure 8(a) and Figure 8(c) as the attack payloads.

C. Discovering Vulnerable Public DNS Resolvers

Before experiments, we test and analyze practical considerations for public resolvers, including the maximum cache TTL, maximum query timeout, and multiple frontend caches introduced in Section V-B.

Maximum cache TTL and query timeout. As listed in Table VIII, for 41 public resolvers, we find that 30 out of 41 support a $\geq$ 1-day TTL, and 28/41 public resolvers' query timeout window is $\geq$ 3 seconds, which provides enough TTL lifetime and attack time window to launch $T1$ and $T2$. We also conduct a measurement of 210k resolvers, and results in Figure 9 show that more than 80% can receive a valid response after nearly 3 seconds, which is sufficient to launch $T1$.

Multiple frontend caches. To evaluate the multiple caches' popularity, through one vantage point, we test and infer the frontend cache number of 41 public DNS resolvers shown in the column of "Cache number" in Table VIII. We send 30 queries to each resolver IP every 5 seconds and use the same source IP address but different UDP ports. Results show that 17 resolvers have more than one frontend cache. We also find that load-balancing policies of 36 resolvers can be bypassed via sending DNS queries with the same TCP/UDP five-tuple[5].

Vulnerable public resolvers. As tabulated in Table VIII (in Appendix C), through repetitious experiments, we show that *all 41 resolver vendors are vulnerable to $T2$ ($T2_A$ and/or $T2_R$)* because they implement current downward delegation mechanism and honor the LSM match algorithm, such as

[5] The five-tuple of TCP/UDP connection is the source IP address, source port, destination IP address, destination port, and transport protocol [75].

TABLE V. DNS RESOLUTION BEHAVIOR DIFFERENCES OF 10 OUT OF 41 POPULAR PUBLIC RESOLVER VENDORS.

Vendor	Vulnerable?				
	G_O	$T1_A$	$T1_R$	$T2_A$	$T2_R$
Google Public DNS [55]	✗	✗	-[1]	✗	✓[2]
Cloudflare DNS [30]	✗	-	-	✓	✓
114DNS [1]	✗	✓	✓	✓	✓
OpenDNS [27]	✗	✓	✓	✓	✓
Level3 DNS [78]	✗	✓	✓	✓	✓
Quad9 DNS [117]	✗	✓	✓	✓	✓
Neustar UltraDNS [99]	✗	✓	✓	✓	✓
Dyn DNS [46]	✗	✓	✓	✓	✓
CleanBrowsing DNS [28]	✗	✓	✓	✓	✓
Yandex.DNS [146]	✗	✓	✓	✓	✓

[1] "-" means the issue has not been identified due to a small timeout value.
[2] Vulnerable if delegating to a new nameserver IP address.
✓: Vulnerable. ✗: Not vulnerable.

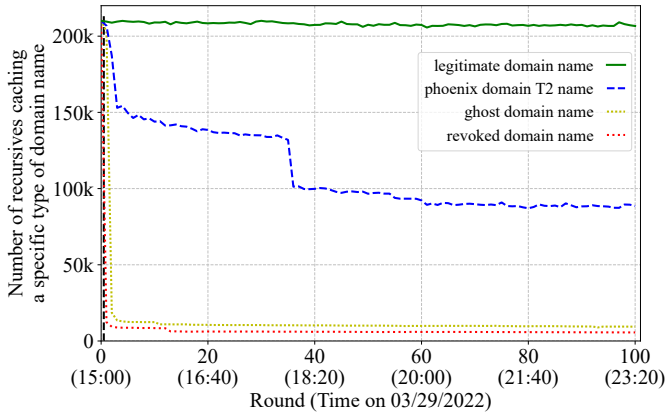


Fig. 10. Resolving of domain names at recursive resolvers over time with a TTL of 10m and a probing interval of 5m. The dashed vertical line indicates the original TTL expiration time of each domain name (test for eight hours).

Google Public DNS [55] and Cloudflare DNS [30]. In the meanwhile, unsurprisingly, they are all immune to the original ghost domain attack since it has been proposed for nearly ten years. 6 resolver vendors are not affected by $T1_A$ and $T2_A$. We extrapolate that they ignore delegation data in the answer response [24], including DNS for Family [40], Google Public DNS [55], DNSPod Public DNS+ [136], CIRA Shield DNS [26], ControlD DNS [35], and UncensoredDNS [142]. For resolvers with a query timeout window of $\geq$ 3s (introduced in Section V-B), we test whether they are vulnerable to $T1$. Results show that resolvers except for CIRA Shield DNS [26] are all *exploitable* by $T1$ ($T1_A$ and/or $T1_R$). Since the implementation of CIRA Shield DNS is not open source, we speculate that it prefers the delegation data from parent zones or aligning the cache use-and-check operation. These mitigation methods will be discussed in Section VII-A. Here, we show 10 out of 41 public resolver vendors in Table V.

D. Detecting Vulnerable Open DNS Resolvers

Measurements and results. Using experiment setups in Section VI-B, we perform large-scale measurements to show the population of exploitable open DNS resolvers by $T2$.

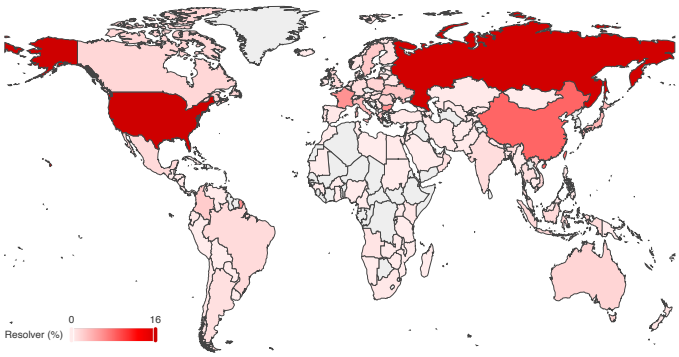


Fig. 11. Geographic view of vulnerable recursive resolvers haunted by Phoenix *T2* domains after 100 rounds for each region and their percentage indicated by different colors.

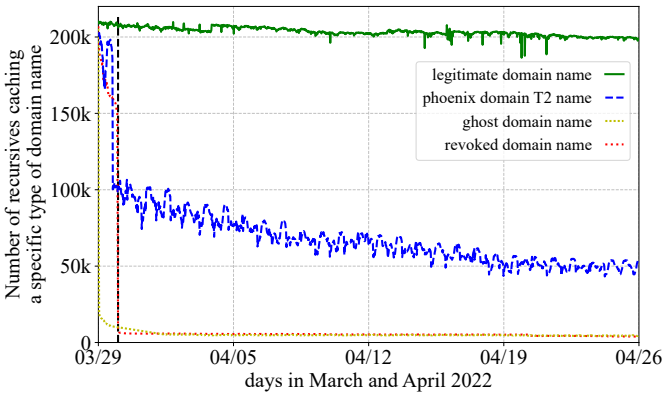


Fig. 12. Resolving of domain names at recursive resolvers over time with a TTL of 1d and a probing interval of 30m. The dashed vertical line indicates the original TTL expiration time of each domain name (test for one month).

Measuring the effect of the iteration number (short-term). Ideally, the more iterative refreshing, the longer cache is maintained. However, since the maximum domain label number is limited by DNS implementations (Section IV-B), attackers can not unlimitedly create a subdomain to perform $T2$ attacks. We first conduct 100 rounds of measurements toward all 210k resolvers to dig into (1) the maximum iterations (i.e., the maximum label number) supported by open resolvers and (2) the possibility of different threat models in real-world networks. For each domain group, we set the TTL to 10 minutes, refresh the DNS cache in a 5-minute interval, and snoop the cache every 5 minutes.

Results in Figure 10 show that more than *187,134* (89%) resolvers return valid responses for the phoenix domain names after the cache has expired, demonstrating all these resolvers are vulnerable to $T2$. Even after 100-round tests, over *88,216* (42%) recursive resolvers are still haunted by $T2$, and we depict the distribution of their geo-location in Figure 11 to show the prevalence in each region. There is a plunge (18.3%) before the 40th test because the maximum label number of Microsoft DNS is *39* confirmed by experiments. Besides, *9,426* (4%) are exploitable for the ghost domain attack. Notice that there are *5,610* (3%) recursives that will never remove DNS cache after a domain is revoked (the red line in Figure 10).

Measuring the longitudinal impact (long-term). Recall that attackers in our threat models aim at exploiting the revoked domain name for malicious activities; thus, the lifetime of PHOENIX DOMAIN is an essential component. To this end, we attempt to conduct a month-long measurement and analyze how many open resolvers can maintain the delegation data continuously. Overall, we set the probing interval to 30 minutes in this measurement and employ a 1-day TTL value. For $T2$, we update the query domain name every 3/4 TTL while setting the TTL value to 1 hour and the refreshing interval to 30 minutes for the ghost domain.

As shown in Figure 12, we find that around *84,015* (40%) recursives vulnerable for $T2$ for at least *one week* while *53,429* (25%) for *one month*, which provides enough time window for malicious domains to be continuously exploitable. The number of vulnerable resolvers falls near the time of cache expiration because real-world factors affect the attack results, as discussed in Section V. Specifically, 40% of resolvers have a maximum cache TTL value of $<$ 1 day (discussed below), and some implementations are immune to the answer payload (listed in Table I). Besides, about *10k* resolvers are not alive after one month and *3,929* (1.4%) still cache the records of revoked domain names, which we will discuss in the following. However, the ghost domain name only last in *2,837* (1.4%) open resolvers after the cache expiration, meaning nearly all resolvers are patched.

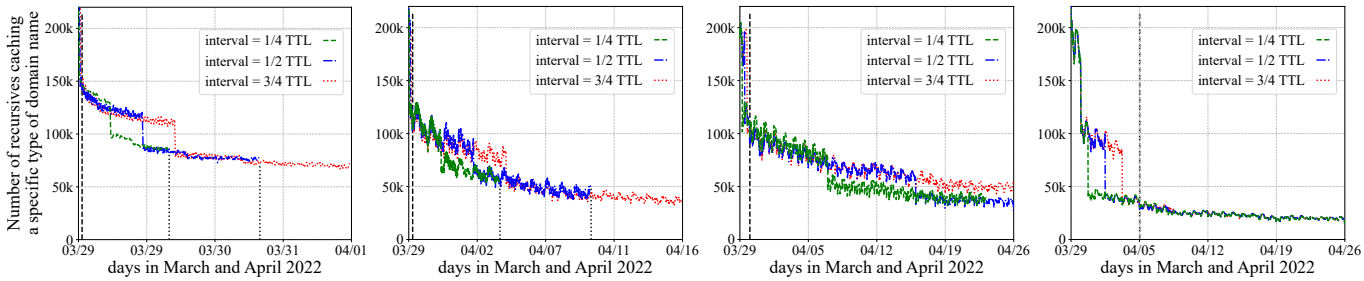
Factors affecting $T2$. Apart from multi-round and long-term measurements, we conduct the following comparative experiments to demonstrate the effects caused by several factors. Different settings can introduce different attack results of $T2$.

Cache TTLs and refreshing intervals. In order to evaluate how TTLs and refreshing intervals influence the possibility of $T2$, we first measure 210k resolvers to see the maximum cache TTL distribution. Results in Figure 14 shows that about 20% are 1 day and $>$ 40% are up to a week. In total, more than 60% have a $>$ 1-day TTL. Surprisingly, *21,546* resolvers have a maximum TTL of $\geq$ 1 year.

Then according to the TTL distribution, we perform measurements on 210k resolvers with TTL values set to 1h, 6h, 1d, and 7d, and refreshing intervals to 1/4 TTL, 1/2 TTL, and 3/4 TTL. From the results illustrated in Figure 13, we find if TTL values are larger, there can be less exploitable resolvers, comparing Figure 13(a) (TTL=1h) and Figure 13(c) (TTL=1d), because most resolvers have a small maximum TTL value shown in Figure 14. In addition, using the same TTL value, we show that the refreshing intervals of the DNS cache may not have much influence on the number of exploitable resolvers. We attribute this to the stable recursive resolvers we used, which cache records following the TTL in a good manner. However, the later the DNS cache is refreshed, the longer a $T2$ domain can survive (the red lines in Figure 13). Attackers can select a sizeable refreshing interval to reduce the attack cost and achieve a similar effect.

Probing times. As discussed in Section V, many DNS resolvers deploy multiple frontend caches[6] for reasons like load balancing, which affects the success rate of cache updates. Considering the average cache number and ethical issues, we compare the results of probing 4 times with 1 time per round.

[6] The average frontend cache number of 41 resolvers (Table VIII) is 4.



(a) TTL = 1h (test for 3 days) (b) TTL = 6h (test for 18 days) (c) TTL = 1d (test for 1 month) (d) TTL = 7d (test for 1 month)

Fig. 13. Resolvers with alive phoenix domain $T2$ names under different TTL values and refreshing intervals setups.

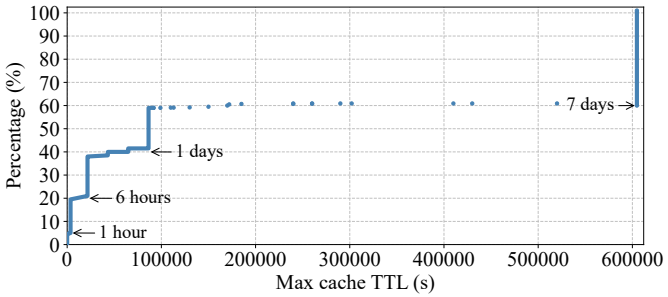


Fig. 14. The fraction of max supported TTL of analyzed recursive resolvers.

Overall, we get extra *3.4%* exploitable resolvers of 210k by probing 4 times. In actual exploitation, attackers can send multiple queries once to increase the success rate.

Other methods to maintain cache. (1) Stale data. Stale data is proposed by RFC 8767 [76] for recursives to serve expired data to provide answers when nameservers cannot be reached. The maximum stale cache lifetime is suggested to be 1 to 3 days. However, as shown in Figure 12 (the red line), we find, after one month, *3,929* resolvers can still successfully resolve the revoked domain name, which do not follow the specifications and can be exploited to inject revoked domains. (2) Large cache TTL. Resolvers tend to overwrite the TTL field to prevent caching records for a long time (up to 68 years). However, as discussed in Section V, *21,546* resolvers have a maximum TTL value of more than one year, and *552* of them still have cached records that we queried one month ago. Administrators should examine their resolvers for these careless misconfigurations or implementations.

E. Ethical Considerations

Our experiments involve many DNS providers, implementations, and open DNS resolvers. Therefore, we incorporate many considerations regarding ethics in the experiment design. We strictly adhere to the existing ethical principles of Menlo Report [70] and best practices of network measurements [108].

First, we conduct controlled experiments. We install DNS software on our machines in the vulnerability validation phase, using the open-sourced mainstream DNS implementations. All domain names and authoritative nameservers are operated by ourselves. Since the domain names are newly created on-demand that no users will access, thus the DNS records we set for them will not influence real-world users. In addition, the testing domain names are mainly "revoked" ones in our threat model; thus, the DNS records cached by open resolvers will be automatically removed after the TTL time. Specifically, for resolvers not purging cached data, we only use them for the *D1* and *D2* experiments and do not insert new records.

Second, for active measurements (e.g., testing public resolvers and probing open resolvers), we strictly limit the scanning rate to reduce the effect on the operation of DNS resolvers [84]. Besides, we restrict the number of DNS queries (up to 4) toward each resolver in each testing round.

Third, we report vulnerabilities to all relevant vendors for responsible disclosure, which we discuss in Section VII-B.

VII. Discussion and Mitigation

A. Mitigation

To defend against PHOENIX DOMAIN attacks, delegation consistency should be guaranteed. Even though DNSSEC has been proposed to validate the delegation trust chain between parent and child zones [15], [16], [17], it does not require a registered domain must be signed and thus can not mitigate PHOENIX DOMAIN. We discuss 6 mitigation approaches according to their effectiveness as listed in Table VI, part of which have been adopted by vendors shown in Section VII-B.

M1: Re-validating delegation information. The root cause of PHOENIX DOMAIN is that the current DNS cache mechanism lacks active cache validating operations. Once records in the upper-level zone are altered or revoked, it takes the TTL time for resolvers to perceive the change. Therefore, we urge resolvers to implement the active cache validating operations like the validation of certificate revocation in the PKI ecosystem [36], [122], especially for the crucial delegation data. For example, resolvers can validate the state of expired `NS` records by requesting the TLD zone, which will eradicate PHOENIX DOMAIN.

This mitigation method has been proposed since 2010 [59], [145], even prior to the original ghost domain attack in 2012, but has not been adopted by implementations until now. Through inspecting the original *M1* document [145] and discussion email lists [126], we found that the community did not come to a common conclusion to re-validate the NS record and apply parent-centric policies [24]. At last, Ghost Domain was mitigated by not allowing TTL extension instead of *M1* (Section II-C). Since then, *M1* was ignored just as an expired

TABLE VI. EFFECTIVENESS SUMMARY OF 6 MITIGATION METHODS.

Mitigation	T1	T2
M1: Re-validating delegation information	●	●
M2: Updating delegation data by parent-centric policies.	●	○
M3: Aligning the cache use-and-check operations	●	○
M4: Ignoring unsolicited DNS records	◐	◐
M5: Scrutinizing domain names with over many labels	○	◐
M6: Restricting the maximum cache TTL	○	◐

●: Fully valid. ◐: Partially valid. ○: Not valid.

draft until 2020, when Huque et al. felt that *M1* was still useful and attempted to propel its standardization [59], [125]. Both they and developers of Unbound [140] did not think *M1* would impose more overhead on implementations.

M2: Updating delegation data by parent-centric policies. As described in Section III, Ghost Domain and $T1$ result from the selfward delegation on authoritative servers of child zones, which return different delegation data on behalf of parent zones. This is a common type of cache inconsistency [106], [131], although DNS specifications require the consistent delegation data between the parent and child zone [94]. To eliminate this inconsistency, resolvers can prioritize the delegation data from the parent zone with a high trust level or only trust delegation information from the upper-level zone. However, it violates current best practices because most implementations tent to trust data from child zones (child-centric) [24]. This measure can defend both Ghost Domain and $T1$, and has been adopted by implements like Technitium and MaraDNS (see Table I).

M3: Aligning the cache use-and-check operations. In the case of the $T1$ attack, the resolver uses cache when it is alive (*time-of-use*) while deleting cache when it is expired (*time-of-check*) without considering the synchronized state that expired data has been used in the current resolution process. Therefore, the inconsistent cache operation between time-of-use and time-of-check during one resolution process results in $T1$. Resolvers should associate the cache use-and-check operations and not remove cached records if the current resolution process uses them. For example, on each query, PowerDNS Recursor deletes expired records when searching for referral data from the cache and does not remove expired data when updating the cache, which alleviates the $T1$ attack.

M4: Ignoring unsolicited DNS records. As shown in Figure 8, attackers leverage the answer response to carry new delegation records in the authority and additional section. Those potential records are not solicited by the queriers and do not pertain to queries. Under the circumstance of child-centric delegation update policies, those unsolicited delegation data can overwrite cached data and facilitate the following queries. However, we believe that resolvers could ignore unsolicited DNS records to mitigate $T1_A$ and $T2_A$, and this operation does not affect performance basically. By modifying the source code of BIND9 and resolving Alexa top 100k domains [9], we test the number of outgoing queries issued by BIND9 with the authority and additional section cached (E_c) and not (E_{nc}). Results show that E_c sends out 250,019 queries while E_{nc} costs 250,269 queries with only a 0.1% increasing.

M5: Scrutinizing domain names with over many labels. When exploiting the $T2$ attack, attackers delegate a new child zone to prolong the total lifetime of cached data. After each downward delegation, $T2$ will be added a label in the front. Therefore, a domain name with too many labels is a distinct characteristic. With the cooperation of a large famous company operating Passive DNS data[7], we find $>$ *99%* of domain names (total *7,703,247* under Alexa top 1M domains [9]) observed in the DNS response log over the past one year have $\leq$ *10* labels. We recommend resolvers check the delegation data of domain names with $>$ 10 labels carefully, such as applying *M1* or *M6*.

M6: Restricting the maximum cache TTL. Since $T2$ leverages 127$\times$ TTL to prolong the total lifetime of cached records, operators can configure a small maximum cache TTL value for their resolvers as a short-term mitigation method. For example, with a 1-hour TTL, the total cache lifetime is no more than 127 hours. In addition, vendors can strengthen the caching mechanism by lowering the cache TTL value of subdomains, such as the TTL of delegation data for the child zone should be less than those from its parent zone.

B. Disclosure and Responses

We have reported vulnerabilities to all relevant vendors for responsible disclosure, including 8 software vendors, 41 public resolver vendors, and other affected vendors, e.g., Akamai DNS services [6] and SDNS [124]. So far, 7 software and 15 public resolver vendors have confirmed the vulnerabilities, and part of them have published the details after negotiation with each other. We are assigned 9 CVE numbers and are still waiting for responses from other vendors. There are months for them to fix those issues before the paper gets published. Below we summarize the discussions and vendors' mitigation plans after they reviewed our report.

BIND: decided that PHOENIX DOMAIN was the DNS protocol problem and opened issues to discuss mitigation *M2* for the development version [24].

Knot: acknowledged the 2 novel attacks and is waiting for the mitigation *M1* draft to be a standard to implement.

Unbound: confirmed the novel PHOENIX DOMAIN attacks and has released a public fix following mitigation *M1* and *M3*, and assigned 2 CVE numbers for 2 vulnerabilities separately [140].

PowerDNS: acknowledged that PHOENIX DOMAIN was a new vulnerability and is considering implementing mitigations.

Microsoft: determined PHOENIX DOMAIN was a generic issue applicable to the DNS protocol and was not specific to it.

Technitium: confirmed the 2 vulnerabilities and patched the update with mitigation *M1* and *M4* [134] after discussion.

MaraDNS: described PHOENIX DOMAIN was a very clever attack and patched MaraDNS with mitigation *M5* and *M6* [88].

Google Public DNS: confirmed the vulnerability in the Bug Hunters platform [54] and awarded us a $500 bounty and is implementing patches.

[7] The Passive DNS data contains captured DNS response log from the public recursive resolver to clients with their approval. Employees in QI-ANXIN Technology Company [115] conducted the label analysis experiment, and no private information was disclosed.

Cloudflare DNS: determined PHOENIX DOMAIN to be a case of DNS poisoning in the HackerOne platform [57] and made security restrictions against it after our report.

AdGuard DNS: replied that PHOENIX DOMAIN seemed to be a "by design" behavior and is discussing mitigation with us.

Akamai DNS Service: affirmed that they are vulnerable to $T2$ but not $T1$, as they implement a separate cache for the record and delegation data. Now they are discussing with us to apply the same separate delegation cache for mitigating $T2$.

114 DNS and AliDNS: confirmed 2 new vulnerabilities. 114 DNS is implementing mitigation *M3* and *M6* as the mitigation approaches, while AliDNS is considering mitigation *M2*.

CNNIC sDNS, DNS For Family, OneDNS, Quad9 DNS, and SDNS: confirmed 2 vulnerabilities and are implementing patches.

360 Secure DNS, Baidu DNS, DNSPod Public DNS, and Dyn DNS: acknowledged the vulnerabilities and are further evaluating the mitigation approaches. Besides, Baidu DNS and DNSPod Public DNS awarded us a bounty respectively.

VIII. RELATED WORK

DNS delegation inconsistency. The significant research effort was dedicated to analyzing issues and root causes of DNS delegation inconsistency. Although RFC 1034 [94] states that the delegation data from parent zones and child zones should be consistent, delegation inconsistencies are prevalent and persistent in practice. Since 2004, Pappas *et al.* [106] has shown the lame delegation affected 15% of the DNS zones, which parent zones pointed to wrong nameservers for child zones. In 2020, Sommese *et al.* [131] delved into TLD and SLD zones and discovered 8% of SLDs (i.e., `.com`, `.net`, and `.org`) exhibit delegation inconsistencies. They proved that the asynchronous delegation would lead to different resolution results among different resolvers. In addition, Akiwate *et al.* [7] demonstrated that DNS misconfigurations could lead to the unauthorized delegation and lame delegation in some zone files affecting 14% of domains they queried. Recently, in 2022, Houser *et al.* [58] found that more than 1,000 domains were vulnerable to hijacking due to defective delegations. Unlike previous research, our work proposed novel PHOENIX DOMAIN attacks, which exploits delegation priorities for referrals from parent and child zones.

DNS Misconfiguration. In 2010, Kalafut *et al.* [69] firstly investigated the orphan DNS server, which had address records in the DNS zone, but the domain it resided in did not exist. Then found that 1.7% of examined DNS servers were orphans, and 1% were used for phishing and malware behaviors. In 2020, Sommese *et al.* [130] re-quantified the orphan record. They found that for `.com` and `.net` TLDs, the number of orphan records had fallen to zero, while for some TLDs, e.g., `.info` and `.mobi`, the number had increased over 10 years. Orphan records result from the zone file misconfiguration maintaining forgotten records in the top-level zone file. In contrast, ten years later, PHOENIX DOMAIN attacks can still manipulate the delegation records continuously resolvable in the cache against the current defense of ghost domain attacks.

DNS Cache Measurement. To analyze the cache mechanism and performance, numerous studies have contributed to DNS cache measurements. In 2013, Schomp *et al.* [123] observed that the TTL value was frequently modified by 64% of resolvers they analyzed, but cache evictions due to capacity limits occurred infrequently. In 2017, Klein *et al.* [72] proposed techniques for cache enumeration and analyzed the complex multiple frontend caches. They utilized a diverse dataset to evaluate their tool from the view of DNS resolution platforms. In 2018, Al-Dalky *et al.* [8] presented a characterization and classification of the multiple recursive resolver pools and showed that pools are dispersed geographically. Moura *et al.* [96] assessed DNS resilience during DDoS attacks and demonstrated DNS caching, retries, and multiple recursives contributed to the attack tolerance. In 2019, Foremski *et al.* [49] analyzed the effect of TTL changes on DNS traffic and found caching a name for a shorter duration led to more queries. Moura *et al.* [97] showed that longer TTLs had significant promise in reducing latency and provided recommendations for configuring TTLs. In 2020, Randall *et al.* [118] leveraged their tool Trufflehunter to infer the caching strategies of four popular public DNS resolvers and estimate domain name usage. In 2022, Niaki *et al.* [100] utilized the TTL value to infer when the resolver cached the records and to localize website filtering appliances. Compared with those works, we also measure the caching and transmission mechanism of DNS software and open resolvers and exploit them to achieve the PHOENIX DOMAIN attacks.

IX. CONCLUSION

Though DNS practices have almost mitigated the threats caused by ghost domain names, the current DNS standards are still flawed for exploiting revoked domains. In this study, we systematically investigate the models of domain name delegation and revocation and mainstream DNS implementations. We propose a general and novel attack, named PHOENIX DOMAIN with two variations, which leverages inconsistent cache use-and-check validation ($T1$ vulnerable to specific implementations) and de facto cache insertion operation ($T2$ applicable to all implementations) to make a revoked domain name continuously resolvable. All DNS implementations can be affected by $T1$ and/or $T2$. To evaluate the real-world implications, we perform large-scale measurements toward 41 public DNS resolvers and 210k open recursives. Results show that all public resolvers and most open recursives can be exploited to launch PHOENIX DOMAIN attacks. For responsible disclosure, we propose 6 feasible suggestions to help the DNS community mitigate the threats. So far, 7 software and 15 resolver vendors have confirmed or fixed the vulnerabilities. 9 CVE-ids have been assigned. We calls for standardization to address the issue of insecure domain name revocation and inconsistent cache.

ACKNOWLEDGEMENT

We thank our shepherd, Doowon Kim, and all the anonymous reviewers for their valuable comments to improve this paper and all software and resolver vendors and our industry partners for their discussion and support. The authors from Tsinghua University were supported by the National Natural Science Foundation of China (U1836213, U19B2034, 62102218, and 62132011). The authors from University of California, Irvine were supported by NSF CNS-2047476 and gifts from Cisco and Microsoft.

REFERENCES

[1] 114DNS, "114DNS," https://www.114dns.com/, 2022.

[2] 360, "360 Secure DNS," https://sdns.360.net/, 2022.

[3] AdGuard DNS, "AdGuard DNS," https://adguard-dns.io/, 2022.

[4] Y. Afek, A. Bremler-Barr, and L. Shafir, "NXNSAttack: Recursive DNS Inefficiencies and Vulnerabilities," in *Proceedings of the 29th USENIX Security Symposium (USENIX Security '20)*, 2020.

[5] AhaDNS, "AhaDNS.com," https://ahadns.com/, 2022.

[6] Akamai, "DNS Infrastructure," https://www.akamai.com/products/dns-infrastructure, 2022.

[7] G. Akiwate, M. Jonker, R. Sommese, I. Foster, G. M. Voelker, S. Savage, and K. Claffy, "Unresolved Issues: Prevalence, Persistence, and Perils of Lame Delegations," in *Proceedings of the ACM Internet Measurement Conference (IMC '20)*, 2020.

[8] R. Al-Dalky and K. Schomp, "Characterization of Collaborative Resolution in Recursive DNS Resolvers," in *Proceedings of the 19th International Conference on Passive and Active Measurement (PAM '18)*, 2018.

[9] Alexa, "The top 1M sites on the web," https://www.alexa.com/, 2022.

[10] AliDNS, "AliDNS," https://alidns.com/, 2022.

[11] E. Alowaisheq, P. Wang, S. Alrwais, X. Liao, X. Wang, T. Alowaisheq, X. Mi, S. Tang, and B. Liu, "Cracking the Wall of Confinement: Understanding and Analyzing Malicious Domain Take-downs," in *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS '19)*, 2019.

[12] O. Alrawi, C. Lever, K. Valakuzhy, R. Court, K. Snow, F. Monrose, and M. Antonakakis, "The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle," in *Proceedings of the 30th USENIX Security Symposium (USENIX Security '21)*, 2021.

[13] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai Botnet," in *Proceedings of the 26th USENIX Security Symposium (USENIX Security '17)*, 2017.

[14] APNIC, "DNS Resolvers Use," https://stats.labs.apnic.net/rvrs, 2022.

[15] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "RFC 4033: DNS Security Introduction and Requirements," *RFC Proposed Standard*, 2005.

[16] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "RFC 4034: Resource Records for the DNS Security Extensions," *RFC Proposed Standard*, 2005.

[17] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "RFC 4035: Protocol Modifications for the DNS Security Extensions," *RFC Proposed Standard*, 2005.

[18] Baidu, "Baidu DNS," https://dudns.baidu.com/, 2022.

[19] BIND, "CVE-2012-1033: Ghost Domain Names: Revoked Yet Still Resolvable," https://kb.isc.org/docs/aa-00691, 2012.

[20] BIND, "How do I change the version that BIND reports when queried for version.bind?" https://kb.isc.org/docs/aa-00359, 2021.

[21] BIND, "BIND 9," https://www.isc.org/bind/, 2022.

[22] BIND, "BIND 9 Configuration Reference," https://bind9.readthedocs.io/en/latest/reference.html, 2022.

[23] BIND, "BIND Source Code Repository," https://gitlab.isc.org/isc-projects/bind9/-/blob/v9_18_2/lib/dns/rbtdb.c#L4880, 2022.

[24] BIND, "Consider parent-centric delegations," https://gitlab.isc.org/isc-projects/bind9/-/issues/3311, 2022.

[25] CenturyLink, "CenturyLink DNS," https://www.centurylink.com/home/help/internet/dns.html, 2022.

[26] CIRA, "CIRA Shield DNS," https://www.cira.ca/cybersecurity-services/canadian-shield, 2022.

[27] Cisco, "OpenDNS," https://www.opendns.com/, 2022.

[28] CleanBrowsing, "CB DNS," https://cleanbrowsing.org/, 2022.

[29] Cloudflare, "Unable to update DDNS using API for some TLDs," https://community.cloudflare.com/t/unable-to-update-ddns-using-api-for-some-tlds/167228, 2020.

[30] CloudFlare, "CloudFlare DNS," https://1.1.1.1/dns/, 2022.

[31] CNNIC sDNS, "CNNIC sDNS," https://www.sdns.cn/, 2022.

[32] E. Cohen and H. Kaplan, "Proactive Caching of DNS Records: Addressing A Performance Bottleneck," in *Proceedings 2001 Symposium on Applications and the Internet (SAINT '01)*, 2001.

[33] Comodo, "DNS," https://www.comodo.com/secure-dns/, 2022.

[34] Comss.one, "DNS," https://www.comss.ru/page.php?id=7315, 2022.

[35] ControlD, "ControlD DNS," https://controld.com/free-dns/, 2022.

[36] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," *RFC Proposed Standard*, 2008.

[37] CZ.NIC, "CZ.NIC ODVR DNS," https://www.nic.cz/odvr/, 2022.

[38] Department of Justice Office of Public Affairs, "United States Leads Seizure..." *The United States Department of Justice*, 2022.

[39] DJB dnscache, "CVE-2012-1191," https://nvd.nist.gov/vuln/detail/CVE-2012-1191, 2012.

[40] DNS for Family, "DNS for Family," https://dnsforfamily.com/, 2022.

[41] DNS Forge, "DNS Forge - DNS Resolver," https://dnsforge.de/, 2022.

[42] DNS-OARC, "Fpdns," https://www.dns-oarc.net/tools/fpdns, 2021.

[43] DNSlify DNS, "DNSlify DNS," https://www.dnslify.com/, 2022.

[44] DNS.WATCH, "DNS.WATCH," https://dns.watch/, 2022.

[45] K. Du, H. Yang, Z. Li, H. Duan, S. Hao, B. Liu, Y. Ye, M. Liu, X. Su, G. Liu, Z. Geng, Z. Zhang, and J. Liang, "TL;DR Hazard: A Comprehensive Study of Levelsquatting Scams," in *Proceedings of the 15th International Conference on Security and Privacy in Communication Systems (SecureComm '19)*, 2019.

[46] Dyn, "Dyn DNS," https://help.dyn.com/internet-guide-setup/, 2022.

[47] R. Elz and R. Bush, "RFC 2181: Clarifications to the DNS Specification," *RFC Proposed Standard*, 1997.

[48] FDN, "FDN DNS," https://www.fdn.fr/actions/dns/, 2022.

[49] P. Foremski, O. Gasser, and G. C. M. Moura, "DNS Observatory: The Big Picture of the DNS," in *Proceedings of the Internet Measurement Conference (IMC '19)*, 2019.

[50] Free DNS, "FreeDNS," https://freedns.zone/, 2022.

[51] Freenom, "DNS," http://www.freenom.world/en/index.html, 2022.

[52] Z. Gao and A. Venkataramani, "Measuring Update Performance and Consistency Anomalies in Managed DNS Services," in *Proceedings of 2019 IEEE Conference on Computer Communications (INFOCOM '19)*, 2019.

[53] GoDaddy, "GoDaddy," https://www.verisign.com/, 2022.

[54] Google, "Bug Hunting," https://bughunters.google.com/, 2022.

[55] Google, "Google Public DNS," https://dns.google/, 2022.

[56] H. Griffioen and C. Doerr, "Examining Mirai's Battle over the Internet of Things," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20)*, 2020.

[57] HackerOne, "HackerOne," https://www.hackerone.com/, 2022.

[58] R. Houser, S. Hao, C. Cotton, and H. Wang, "A Comprehensive, Longitudinal Study of Government DNS Deployment at Global Scale," in *Proceedings of the 2022 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '22)*, 2022.

[59] S. Huque, P. Vixie, and R. Dolmans, "Draft: Delegation Revalidation by DNS Resolvers," *RFC Draft*, 2022.

[60] Hurricane Electric, "HE DNS," https://dns.he.net/, 2022.

[61] ICANN, "Uniform Domain Name Dispute Resolution Policy," https://www.icann.org/resources/pages/policy-2012-02-25-en, 1999.

[62] ICANN, "Guidance for Preparing Domain Name Orders, Seizures & Takedowns," https://www.icann.org/en/system/files/files/guidance-domain-seizures-07mar12-en.pdf, 2012.

[63] ICANN, "2013 Registrar Accreditation Agreement," https://www.icann.org/resources/pages/approved-with-specs-2013-09-17-en, 2013.

[64] ICANN, "Domain Abuse Activity Reporting (DAAR) System," https://www.icann.org/en/system/files/files/daar-monthly-report-30apr22-en.pdf, 2022.

[65] L. Izhikevich, R. Teixeira, and Z. Durumeric, "LZR: Identifying Unexpected Internet Services," in *Proceedings of the 30th USENIX Security Symposium (USENIX Security '21)*, 2021.

[66] P. Jeitner and H. Shulman, "Injection Attacks Reloaded: Tunnelling Malicious Payloads over DNS," in *Proceedings of the 30th USENIX Security Symposium (USENIX Security '21)*, 2021.

[67] J. Jiang, J. Liang, K. Li, J. Li, H.-X. Duan, and J. Wu, "Ghost Domain Names: Revoked Yet Still Resolvable," in *Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS '12)*, 2012.

[68] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS Performance and the Effectiveness of Caching," *IEEE/ACM Transactions on Networking*, 2002.

[69] A. J. Kalafut, M. Gupta, C. A. Cole, L. Chen, and N. E. Myers, "An Empirical Study of Orphan DNS Servers in the Internet," in *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference (IMC '10)*, 2010.

[70] E. Kenneally and D. Dittrich, "The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research," *SSRN Electronic Journal*, 2012.

[71] A. Klein, "Cross Layer Attacks and How to Use Them (for DNS Cache Poisoning, Device Tracking and More)," in *Proceedings of 2020 IEEE Symposium on Security and Privacy (S&P '21)*, 2021.

[72] A. Klein, H. Shulman, and M. Waidner, "Counting in the Dark: DNS Caches Discovery and Enumeration in the Internet," in *Proceedings of the 2017 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '17)*, 2017.

[73] Knot, "Doc," https://knot-resolver.readthedocs.org/en/stable, 2022.

[74] Knot Resolver, "Knot Resolver," https://www.knot-resolver.cz/, 2022.

[75] M. V. Larsen and F. Gont, "RFC 6056: Recommendations for Transport-Protocol Port Randomization," *RFC Best Current Practice*, 2011.

[76] D. C. Lawrence, W. A. Kumari, and P. Sood, "RFC 8767: Serving Stale Data to Improve DNS Resiliency," *RFC Proposed Standard*, 2020.

[77] H. Lee, A. Gireesh, R. v. Rijswijk-Deij, T. T. Kwon, and T. Chung, "A Longitudinal and Comprehensive Study of the DANE Ecosystem in Email," in *Proceedings of the 29th USENIX Security Symposium (USENIX Security '20)*, 2020.

[78] Level3, "DNS," https://www.publicdns.xyz/public/level3.html, 2022.

[79] J. Leyden, "Microsoft seizes Chinese dot-org..." *The Register*, 2012.

[80] X. Li, B. Liu, X. Zheng, H. Duan, Q. Li, and Y. Huang, "Fast IPv6 Network Periphery Discovery and Security Implications," in *Proceedings of the 2021 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '21)*, 2021.

[81] X. Li, C. Lu, B. Liu, Q. Zhang, Z. Li, H. Duan, and Q. Li, "The Maginot Line: Attacking the Boundary of DNS Caching Protection," in *Proceedings of the 32nd USENIX Security Symposium (USENIX Security '23)*, 2023.

[82] LibreDNS, "LibreDNS," https://libredns.gr/, 2022.

[83] B. Liu, C. Lu, H.-X. Duan, Y. Liu, Z. Li, S. Hao, and M. Yang, "Who Is Answering My Queries: Understanding and Characterizing Interception of the DNS Resolution Path," in *Proceedings of the 27th USENIX Security Symposium (USENIX Security '18)*, 2018.

[84] K. Man, Z. Qian, Z. Wang, X. Zheng, Y. Huang, and H. Duan, "DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20)*, 2020.

[85] K. Man, X. Zhou, and Z. Qian, "DNS Cache Poisoning Attack: Resurrections with Side Channels," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21)*, 2021.

[86] J. Mao, M. Rabinovich, and K. Schomp, "Assessing Support for DNS-over-TCP in the Wild," in *Proceedings of the 23rd International Conference on Passive and Active Measurement (PAM '22)*, 2022.

[87] MaraDNS, "MaraDNS," https://maradns.samiam.org/, 2022.

[88] MaraDNS, "MaraDNS release 3.5.0022," https://maradns.samiam.org/security.html#CVE-2022-30256, 2022.

[89] MaxMind, "GeoLite2 Free Geolocation Data," https://dev.maxmind.com/geoip/geolite2-free-geolocation-data, 2022.

[90] MediaNama, "Restriction on bulk domain registrations is for national security: NIXI," https://www.medianama.com/2022/02/223-nixi-bulk-domain-registrations-reason/, 2022.

[91] X. Mi, X. Feng, X. Liao, B. Liu, X. Wang, F. Qian, Z. Li, S. Alrwais, L. Sun, and Y. Liu, "Resident Evil: Understanding Residential IP Proxy as a Dark Service," in *Proceedings of 2019 IEEE Symposium on Security and Privacy (S&P '19)*, 2019.

[92] Microsoft, "LockingPercent," https://docs.microsoft.com/en-us/powershell/module/dnsserver/set-dnsservercache, 2022.

[93] Microsoft DNS, "Domain Name System (DNS) Docs," https://docs.microsoft.com/en-us/windows-server/networking/dns/dns-top, 2022.

[94] P. V. Mockapetris, "RFC 1034: Domain Names - Concepts and Facilities," *RFC Standard*, 1987.

[95] P. V. Mockapetris, "RFC 1035: Domain Names - Implementation and Specification," *RFC Standard*, 1987.

[96] G. C. M. Moura, J. Heidemann, M. Müller, R. d. O. Schmidt, and M. Davids, "When the Dike Breaks: Dissecting DNS Defenses During DDoS," in *Proceedings of the Internet Measurement Conference 2018 (IMC '18)*, 2018.

[97] G. C. M. Moura, J. Heidemann, R. d. O. Schmidt, and W. Hardaker, "Cache Me If You Can: Effects of DNS Time-to-Live," in *Proceedings of the Internet Measurement Conference (IMC '19)*, 2019.

[98] Neustar, "Neustar Announces Acquisition of Verisign's Public DNS Service," https://www.home.neustar/about-us/news-room/press-releases/2020/neustar-announces-acquisition-of-verisigns-public-dns-service, 2020.

[99] Neustar, "UltraDNS Public," https://www.publicdns.neustar/, 2022.

[100] A. A. Niaki, W. R. Marczak, S. Farhoodi, A. McGregor, P. Gill, and N. Weaver, "Cache Me Outside: A New Look at DNS Cache Probing," in *Proceedings of the 22nd International Conference on Passive and Active Measurement (PAM '21)*, 2021.

[101] No-IP, "No-IP: Free Dynamic DNS," https://www.noip.com/, 2022.

[102] Norton DNS, "Norton DNS," https://nortondns.com/, 2022.

[103] A. Oest, Y. Safaei, P. Zhang, B. Wardman, K. Tyers, Y. Shoshitaishvili, and A. Doupé, "PhishTime: Continuous Longitudinal Measurement of the Effectiveness of Anti-phishing Blacklists," in *Proceedings of the 29th USENIX Security Symposium (USENIX Security '20)*, 2020.

[104] A. Oest, P. Zhang, B. Wardman, E. Nunes, J. Burgis, A. Zand, K. Thomas, A. Doupé, and G.-J. Ahn, "Sunrise to Sunset: Analyzing the End-to-end Life Cycle and Effectiveness of Phishing Attacks at Scale," in *Proceedings of the 29th USENIX Security Symposium (USENIX Security '20)*, 2020.

[105] OneDNS, "OneDNS," https://onedns.net/, 2022.

[106] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, and L. Zhang, "Impact of Configuration Errors on DNS Robustness," in *Proceedings of the 2004 Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM '04)*, 2004.

[107] J. Park, D. Nyang, and A. Mohaisen, "Timing is almost everything: Realistic evaluation of the very short intermittent ddos attacks," in *16th Annual Conference on Privacy, Security and Trust (PST '18)*, 2018.

[108] C. Partridge and M. Allman, "Ethical Considerations in Network Measurement Papers," *Communications of the ACM*, 2016.

[109] P. Pearce, B. Jones, F. Li, R. Ensafi, N. Feamster, N. Weaver, and V. Paxson, "Global Measurement of DNS Manipulation," in *Proceedings of the 26th USENIX Security Symposium (USENIX Security '17)*, 2017.

[110] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A Comprehensive Measurement Study of Domain Generating Malware," in *Proceedings of the 25th USENIX Security Symposium (USENIX Security '16)*, 2016.

[111] J. Postel, "RFC 1591: Domain Name System Structure and Delegation," *Informational*, 1994.

[112] PowerDNS, "CVE-2012-1193," https://nvd.nist.gov/vuln/detail/CVE-2012-1193, 2012.

[113] PowerDNS, "Doc," https://doc.powerdns.com/recursor/, 2022.

[114] PowerDNS, "PowerDNS," https://www.powerdns.com/, 2022.

[115] QI-ANXIN, "QI-ANXIN Technology," http://en.qianxin.com/, 2022.

[116] Quad101 DNS, "DNS," https://101.101.101.101/index_en.html, 2022.

[117] Quad9 DNS, "Quad9 DNS," https://www.quad9.net/, 2022.

[118] A. Randall, E. Liu, G. Akiwate, R. Padmanabhan, G. M. Voelker, S. Savage, and A. Schulman, "Trufflehunter: Cache Snooping Rare Domains at Large Public DNS Resolvers," in *Proceedings of the ACM Internet Measurement Conference (IMC '20)*, 2020.

[119] Safe Surfer, "Safe Surfer DNS," https://safesurfer.io/, 2022.

[120] SafeDNS, "SafeDNS," https://www.safedns.com/, 2022.

[121] Sam Trenholme, "The ghost domain bug," https://samiam.org/blog/20120314.html, 2012.

[122] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "RFC 6960: X. 509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP," *RFC Proposed Standard*, 2013.

[123] K. Schomp, T. Callahan, M. Rabinovich, and M. Allman, "On measuring the client-side DNS infrastructure," in *Proceedings of the 2013 Internet Measurement Conference (IMC '13)*, 2013.

[124] SDNS, "Recursive Resolver," https://github.com/semihalev/sdns, 2022.

[125] Shumon Huque, "Delegation Revalidation by DNS Resolvers," https://www.huque.com/2020/05/03/ns-revalidation.html, 2020.

[126] Shumon Huque, "[dns-operations] OpenDNS, Google, Nominet - New delegation update failure mode," https://lists.dns-oarc.net/pipermail/dns-operations/2020-April/020041.html, 2020.

[127] R. D. Silva, M. Nabeel, C. Elvitigala, I. Khalil, T. Yu, and C. Keppitiyagama, "Compromised or Attacker-Owned: A Large Scale Classification and Study of Hosting Domains of Malicious URLs," in *Proceedings of the 30th USENIX Security Symposium (USENIX Security '21)*, 2021.

[128] Simple DNS Plus, "DNS," https://simpledns.plus/download, 2022.

[129] SkyDNS, "SkyDNS," https://www.skydns.ru/, 2022.

[130] R. Sommese, M. Jonker, R. v. Rijswijk-Deij, A. Dainotti, K. Claffy, and A. Sperotto, "The Forgotten Side of DNS: Orphan and Abandoned Records," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW '20)*, 2020.

[131] R. Sommese, G. C. M. Moura, M. Jonker, R. v. Rijswijk-Deij, A. Dainotti, K. C. Claffy, and A. Sperotto, "When Parents and Children Disagree: Diving into DNS Delegation Inconsistency," in *Proceedings of the 21st International Conference on Passive and Active Measurement (PAM '20)*, 2020.

[132] S. Son and V. Shmatikov, "The Hitchhiker's Guide to DNS Cache Poisoning," in *Proceedings of the 6th Itenational ICST Conference on Security and Privacy in Communication Systems (SecureComm '10)*, 2010.

[133] Strongarm DNS, "Strongarm DNS," https://strongarm.io/, 2022.

[134] Technitium, "Change log v8.1," https://github.com/TechnitiumSoftware/DnsServer/blob/master/CHANGELOG.md#version-81, 2022.

[135] Technitium, "Technitium DNS," https://technitium.com/dns/, 2022.

[136] Tencent, "DNSPod Public DNS+," https://www.dnspod.com/, 2022.

[137] Unbound, "Fix ghost domain," https://www.nlnetlabs.nl/projects/unbound/download/#unbound-1-4-17, 2012.

[138] Unbound, "Ghost domain names attack," https://www.nlnetlabs.nl/projects/unbound/security-advisories/, 2012.

[139] Unbound, "About," https://nlnetlabs.nl/projects/unbound/about/, 2022.

[140] Unbound, "Novel "ghost domain names" attacks," https://www.nlnetlabs.nl/projects/unbound/security-advisories/, 2022.

[141] Unbound, "Unbound Doc," https://unbound.docs.nlnetlabs.nl/, 2022.

[142] UncensoredDNS, "DNS," https://blog.uncensoreddns.org/, 2022.

[143] Verisign, "Verisign Public DNS," https://www.publicdns.xyz/public/verisign.html, 2015.

[144] Verisign, "Domain Name Registry," https://www.verisign.com/, 2022.

[145] P. Vixie, R. Joffe, and F. A. C. Neves, "Draft: Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness," *RFC Draft*, 2010.

[146] Yandex, "Yandex.DNS," https://dns.yandex.com/, 2022.

[147] X. Zheng, C. Lu, J. Peng, Q. Yang, D. Zhou, B. Liu, K. Man, S. Hao, H. Duan, and Z. Qian, "Poison Over Troubled Forwarders: A Cache Poisoning Attack Targeting DNS Forwarding Devices," in *Proceedings of the 29th USENIX Security Symposium (USENIX Security '20)*, 2020.

APPENDIX

A. DNS Software Analysis

TABLE VII. ANALYSIS WORKLOAD ON DNS SOFTWARE. "# DAYS" IS HOW LONG WE SPENT ON SOFTWARE ANALYSIS. ALL ANALYZED VERSIONS ARE CURRENT DURING OUR ANALYSIS PERIOD (APRIL, 2022).

Software	# Key SLOC	# Files	# Days
BIND9 [21]	360,178 (C)	1,558	3
Knot Resolver [74]	36,941 (C)	646	2
Unbound [139]	129,695 (C)	430	2
PowerDNS Recursor [114]	86,773 (C++)	387	2
Microsoft DNS [93]	-	-	0.5
Simple DNS Plus [128]	-	-	0.5
Technitium [135]	35,656 (C#)	181	1
MaraDNS [87]	54,748 (C)	1,414	1
Total	**703,991**	**4,616**	**12**

B. Resolver Cache Search Process

Algorithm 1: Simplified pseudo-code of BIND9 cache searching[7].

```
input : db (the cache database)
        qname (queried domain name)
        qtype (queried domain type)
output: DNS data
cache_find()
    // Search down from the root of the db tree.
    // Compare the name label by label from right to left.
    node = find_node(db, qname)
    if (node == NULL) then
        goto find_ns
    // Look for data with the exact type in the node.
    data = find_data(node, qname, qtype)
    if (data == NULL) then
        goto find_ns
    return data
    find_ns: // Find the closest nameserver by LSM.
    NS = find_closest_ns(db, qname)
    return NS
find_closest_ns(db, qname)
    while (qname != ROOT) do
        node = find_node(db, qname)
        if (node != NULL) then
            // Look for data of NS type in the node.
            NS_closest = find_data(node, qname, NS)
            if (data != NULL) then
                return NS_closest
        qname = remove the leftmost label of qname
    return NS_root
```

C. Public DNS Resolver

Table VIII lists all evaluated 41 popular public resolver vendors and the differences in their DNS resolving behaviour.

[7] Summarized from the file bind9/lib/dns/rbtdb.c#L4880 [23].

TABLE VIII. DNS RESOLUTION BEHAVIOR DIFFERENCES OF 41 POPULAR PUBLIC RESOLVER VENDORS

Vendor	Resolver IP	Max TTL	Max timeout	Query count	Cache number[1]	Bypass load balancing[2]	DNSSEC	Vulnerable?				
								G_O	$T1_A$	$T1_R$	$T2_A$	$T2_R$
OneDNS [105]	117.50.10.10	5m	10s	20	1	✓	✓	✗	✓	✓	✓	✓
360 Secure DNS [2]	101.226.4.6	30m	5s	3	3	✓	✗	✗	✓	✓	✓	✓
Ali DNS [10]	223.5.5.5	1h	1s	2	4	✓	✗	✗	✓	✓	✓	✓
114DNS [1]	114.114.114.114	1h	10s	5	1	✓	✗	✗	✓	✓	✓	✓
Quad101 DNS [116]	101.101.101.101	4h	10s	6	1	✓	✓	✗	✓	✓	✓	✓
DNS for Family [40]	78.47.64.161	6h	1s	1	1	✓	✗	✗	✗	✓	✗	✓
Google Public DNS [55]	8.8.8.8	6h	1s	1	19	✗	✓	✗	✗	-[3]	✗	✓[4]
Neustar UltraDNS [99]	156.154.70.1	12h	10s	8	4	✓	✓	✗	✓	✓	✓	✓
Verisign Public DNS [143][5]	64.6.64.6	12h	10s	8	4	✓	✓	✗	✓	✓	✓	✓
Quad9 DNS [117]	9.9.9.9	12h	14s	7	5	✗	✓	✗	✓	✓	✓	✓
Norton DNS [102]	199.85.126.10	12h	25s	9	4	✓	✓	✗	✓	✓	✓	✓
Yandex.DNS [146]	77.88.8.1	<1d[6]	15s	8	22	✗	✓	✗	✓	✓	✓	✓
CleanBrowsing DNS [28]	185.228.168.10	1d	1.5s	1	1	✓	✓	✗	✓	✓	✓	✓
DNS Forge [41]	176.9.1.117	1d	1.5s	1	4	✓	✓	✗	-	-	✓	✓
DNSlify DNS [43]	185.235.81.1	1d	1.5s	1	1	✓	✓	✗	-	-	✓	✓
FreeDNS [50]	37.235.1.174	1d	1.5s	1	5	✓	✓	✗	-	-	✓	✓
Hurricane Electric DNS [60]	74.82.42.42	1d	1.5s	1	1	✓	✓	✗	-	-	✓	✓
LibreDNS [82]	88.198.92.222	1d	1.5s	1	1	✓	✓	✗	-	-	✓	✓
Safe Surfer DNS [119]	104.155.237.225	1d	1.5s	1	1	✓	✓	✗	-	-	✓	✓
Strongarm DNS [133]	52.3.100.184	1d	1.5s	1	1	✓	✓	✗	-	-	✓[7]	✓
AdGuard DNS [3]	94.140.14.14	1d	10s	10	1	✓	✓	✗	✓	✓	✓	✓
Dyn DNS [46]	216.146.35.35	1d	10s	8	2	✓	✓	✗	✓	✓	✓	✓
Freenom World DNS [51]	80.80.80.80	1d	10s	7	15	✗	✓	✗	✓	✓	✓	✓
AhaDNS [5]	185.213.26.187	1d	20s	1	1	✓	✓	✗	✓	✓	✓	✓
DNSPod Public DNS+ [136]	119.28.28.28	1d	20s	34	2	✓	✗	✗	✗	✓	✗	✓
FDN DNS [48]	80.67.169.12	1d	25s	8	1	✓	✓	✗	✓	✓	✓	✓
SafeDNS [120]	195.46.39.39	1d	60s	8	1	✓	✓	✗	✓	✓	✓	✓
SkyDNS [129]	193.58.251.251	1d	60s	8	1	✓	✓	✗	✓	✓	✓	✓
Comodo Secure DNS [33]	8.20.247.10	1d	300s	2	1	✓	✓	✗	✓	✓	✓	✓
CZ.NIC ODVR DNS [37]	185.43.135.1	6d	0.5s	1	1	✓	✓	✗	-	-	✓	✓
CIRA Shield DNS [26]	149.112.121.10	7d	2s	2	1	✓	✓	✗	✗	✗	✗	✓
OpenDNS [27]	208.67.220.120	7d	5s	5	16	✓	✓	✗	✓	✓	✓	✓
Baidu DNS [18]	180.76.76.76	7d	8s	2	10	✓	✗	✗	✓	✓	✓	✓
CenturyLink DNS [25]	205.171.2.26	7d	10s	6	1	✓	✓	✗	✓	✓	✓	✓
ControlD DNS [35]	76.76.2.0	7d	10s	6	1	✓	✓	✗	✗	✓	✗	✓
UncensoredDNS [142]	89.233.43.71	7d	10s	5	1	✓	✓	✗	✗	✓	✗	✓
CNNIC sDNS [31]	1.2.4.8	7d	80s	20	2	✓	✓	✗	✓	✓	✓	✓
Cloudflare DNS [30]	1.1.1.1	$2^{32}-1$s	2s	2	6	✓	✓	✗	-	-	✓	✓
Level3 DNS [78]	4.2.2.1	$2^{31}-1$s	6s	6	2	✓	✓	✗	✓	✓	✓	✓
DNS.WATCH [44]	84.200.69.80	$2^{31}-1$s	10s	10	1	✓	✓	✗	✓	✓	✓	✓
Comss.one DNS [34]	93.115.24.204	$2^{31}-1$s	50s	14	1	✓	✓	✗	✓	✓	✓	✓

[1] Tested and inferred from one vantage point.
[2] Bypassing the load balancing policy with same five tuples <*Src-IP, Dst-IP, Src-Port, Dst-Port, Protocol*> for each DNS query.
[3] "-" means that the vulnerability has not been identified yet due to the small timeout value.
[4] Vulnerable if delegating to a new nameserver IP address.
[5] Acquired by Neustar UltraDNS [98].
[6] A random TTL value will be returned for each query.
[7] Vulnerable if delegating to a new `NS` name different from the queried domain name.
✓: Yes. ✗: No. ✓: Vulnerable. ✗: Not vulnerable.