USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

# Misty Registry: An Empirical Study of Flawed Domain Registry Operation

Mingming Zhang, *Zhongguancun Laboratory;* Yunyi Zhang, *National University of Defense Technology and Tsinghua University;* Baojun Liu and Haixin Duan, *Tsinghua University and Zhongguancun Laboratory;* Min Zhang, Fan Shi, and Chengxi Xu, *National University of Defense Technology*

https://www.usenix.org/conference/usenixsecurity25/presentation/zhang-mingming

This paper is included in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

# Misty Registry: An Empirical Study of Flawed Domain Registry Operation

Mingming Zhang [§]*, Yunyi Zhang [†‡]* , Baojun Liu[‡§]✉, Haixin Duan[‡§]
Min Zhang[†]✉ , Fan Shi[†], Chengxi Xu[†]
[†]National University of Defense Technology, [§]Zhongguancun Laboratory, [‡]Tsinghua University
{zhangyyzyy, zhangmindy, shifan17, xuchengxi}@nudt.edu.cn
{zhangmm}@mail.zgclab.edu.cn, {lbj, duanhx}@tsinghua.edu.cn,

## Abstract

Domain registries manage the entire lifecycle of domain names within TLDs and interact with domain registrars through the Extensible Provisioning Protocol (EPP) specification. Although they adhere to standard policies, EPP implementations and operational practices can vary between registries. Even minor operational flaws at registries can expose their managed resources to abuse. However, registry operations' closed and opaque nature has limited understanding of these practices and their potential threats. In this study, we systematically analyzed the security of EPP operations across TLD registries. By analyzing the entire domain lifecycle and mapping operations to corresponding domain statuses, we discovered that registry operations are attributed to overlapping statuses and complex triggering factors. To uncover flaws in registry operations, we employed diverse data sources, including TLD zone files, historical domain registration data, and real-time registrar interfaces for comprehensive domain statuses. The analysis combined static and dynamic techniques, allowing us to externally assess domain existence and registration status, thereby revealing the inner workings of registry policies. Eventually, we discovered three novel EPP implementation deficiencies that pose domain abuse risks in major registries, including Identity Digital, Google, and Nominet. Evidence has shown that adversaries are covertly exploiting these vulnerabilities. Our experiments reveal that over 1.6 million domain names, spanning more than 50% of TLDs (e.g., `.app` and `.top`), are vulnerable due to these flawed operations. To address these issues, we responsibly disclosed the problem to the affected registries and assisted in implementing a solution. We believe that these registry operation issues require increased attention from the community.

* Both authors contributed equally to this work.
✉ Corresponding authors.

## 1 Introduction

Domain names are vital for identifying and addressing key Internet services. Their lifecycles are maintained by domain registrars and registries through the interfaces of the Extensible Provisioning Protocol (EPP) [27–29]. This protocol establishes rules for domain creation, management, and deletion. When registrars issue EPP commands, domain registries are responsible for processing the relevant Top-Level Domain (TLD) zone files and managing domain resources, such as domain statuses and delegation information. The reality is that independent EPP implementations among different registries require registrars to adapt to these disparate interfaces.

An in-depth security exploration of EPP implementations and operation strategies across domain registries remains critical for preventing domain abuses but is by no means trivial. Existing research [16] has highlighted insecure practices of EPP implementation at the registrar level; namely, renaming host objects during domain deletion can pose domain hijacking risks. Additionally, flaws in EPP server deployments can expose TLD zones to significant security threats [50]. However, the opaque nature of EPP implementations hinders the community's understanding of registry operations and potential vulnerabilities. *Hence, we aim to conduct an empirical study to explore vulnerable operations in domain registries.*

**Challenges in exploring registry operations.** Managing the domain lifecycle involves various phases (e.g., creation, configuration, deletion) and components (e.g., WHOIS servers, zone files). While ICANN guidelines [1, 10, 11] and EPP standards [27–29] outline domain management policies, there is a lack of public disclosure of TLD registry practices. Meanwhile, EPP implementation in terms of specific registries is opaque and complex. Different registries tend to implement their own EPP interfaces independently, with access restricted to accredited registrars. To this end, studying security flaws in registry operations from an external view is challenging.

**Our study.** We conducted a systematic security study of domain registry operations, focusing on identifying potential attack surfaces at the registry level. To achieve this, we empir-

ically analyzed the entire domain lifecycle, mapping operations to relevant domain statuses. We revealed that the domain lifecycle managed by registries is significantly more complex than that analyzed at the registrar level [36]. From the registry's perspective, managing a domain through its lifecycle involves overlapping domain statuses and intricate triggering factors (e.g., ICANN policies, registrar or registrant settings).

To this end, we designed a combined dynamic and static method to assess domain existence, registration statuses, and related registry operations. This approach leveraged extensive datasets, which include over 7.9 billion historical WHOIS records, zone files from 1,109 TLDs, and bulk registration interfaces from registrars. Using these datasets, we analyzed longitudinal operational results and nameserver settings. We also tracked zone file updates linked to registrant behaviors to investigate registries' management models of DNS objects.

Our empirical study revealed that various insecure registry practices, even those intended as protective measures, can expose unregistered or ownerless domains to abuse risks throughout the domain lifecycle: (1) During the ***domain creation process***, some registries protect domain brands or prevent domain abuse by creating additional domain objects, which we term as "twin domain names". However, this practice can unintentionally introduce new attack surfaces for unregistered Internationalized Domain Names (IDNs). We identified 6K IDNs that are vulnerable to twin domain takeover threats or have already been exploited. (2) During the ***domain management process***, we defined two DNS host management practices and observed significant issues with siloed host object management. Some registries automatically add all registrant-configured host objects to zone files without checks or purging, leading to over 80K exploitable stale glue records that affect around 1.6M domains. (3) During the ***domain deletion process***, we identified flawed operations that can lead to domain hijacking. Specifically, registries reset the lifecycle statuses of expired domains to "unregistered" but fail to purge their delegation records from TLD zone files. We termed these "relic domains" and showed that unpurged delegation records make them exploitable for abuse. Ultimately, we identified 3.4K relic domains and 6 vulnerable registry backends (e.g., ZDNS, GoDaddy, and Identity Digital) that supported 812 TLDs, such as `.app` and `.dev`, highlighting a widespread issue among domain registries.

Our research shows these threats are realistic: we detected actual domain abuse cases where adversaries covertly exploit ownerless domains for monetization or market promotion. We responsibly disclosed the issues to the affected registries and their backend service providers and received positive responses. Two registry backends, ZDNS (e.g., `.top`) and Nomulus (e.g., `.app`), responded promptly to our disclosure and timely resolved these issues. Moreover, we are also reaching out to more registries and service providers through the ICANN community to raise awareness of the flaws in EPP implementations and registry operations.

**Contributions.** We make the following contributions:

- We conduct a top-down empirical security study at the domain registry level. We systematically examine domain registry operations across the entire domain lifecycle, uncovering flawed operations that introduce domain abuse threats.
- We perform a large-scale measurement study utilizing multifaceted public datasets and evaluate the real-world security impact of the flawed operations.
- We responsibly disclose the threat to relevant domain registries, provide practical detection and mitigation guidelines, and have received confirmation from a major domain registry.

## 2 Background

### 2.1 Domain Registration and Management

**Domain registration.** Domain names are registered and managed by accredited registry operators (e.g., VeriSign) and registrars (e.g., GoDaddy, Namecheap). *Registries* operate the Top-Level Domain (TLD, e.g., `.com`) zones and manage registration data and delegation records (e.g., NS and glue records) for all domain names within their TLDs. These registries delegate domain registration services to *registrars*, which in turn sell Second-Level Domains (SLDs, e.g., `foo.com`).

A *registrant* can register a domain name from a registrar by submitting registration information and configuring delegation records. Upon receiving the request, the registrar checks the domain's availability and validity. When the checks are passed, the registrar submits the domain object to the corresponding TLD zone through registry-provided interfaces. The registry then processes the request, creates domain objects, and updates TLD zone files. Notably, the registration information and domain status are maintained by both the registry and the registrar using WHOIS [21] or RDAP [4].

**Domain name and delegation host management.** Registrants can choose from three types of authoritative servers (i.e., nameservers or NSes) to configure delegation hosts for their domain names: 1) *Registrar NS*: Default nameservers provided by registrars. 2) *On-premise NS*: Custom nameservers managed by registrants themselves. 3) *Cloud-hosted NS*: Public hosting services like Amazon Route 53 or NSONE, offering specialized and secure domain resolution services. In the first and third scenarios, registrants rely on providers' nameservers, with the configured NS records written into TLD zone files. In contrast, the second option requires storing additional delegation records, known as *glue records*, in the TLD zone files to prevent resolution loops [42]. For example, if `foo.example`'s NS is `ns.foo.example`, a recursive resolver must query `foo.example` to resolve `ns.foo.example`, causing a resolution loop. Thus, registrants should register a DNS host object (e.g., `ns.foo.example A 1.2.3.4`) and save it in the TLD zone file as a glue record. The record is included in referral responses to assist with domain resolution.

To improve DNS transparency, ICANN provides public access to all delegation records, including NS and glue records

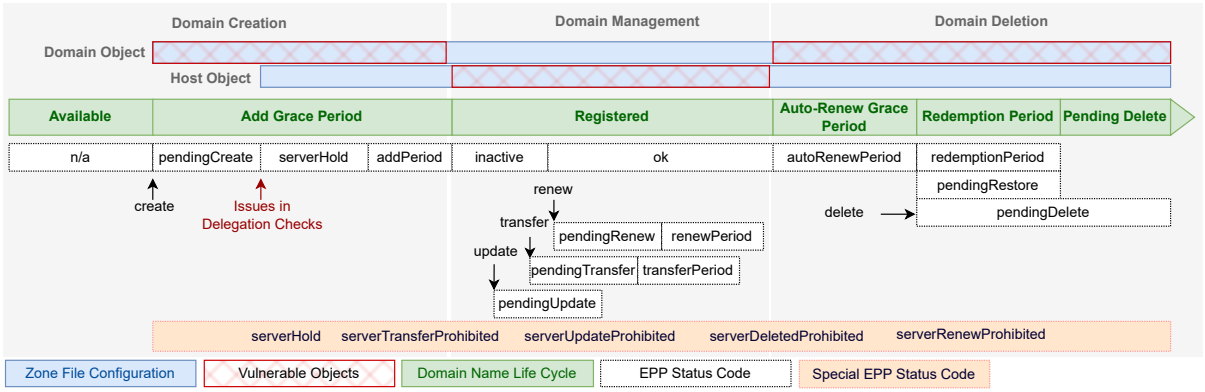


Figure 1: The domain lifecycle stages with corresponding EPP statuses and registry operations. "Vulnerable Objects" reflect potentially vulnerable stages identified in our analysis.

in TLD zone files, via the Centralized Zone Data Service (CZDS) [7], which is essential for domain management. Additionally, some registries outsource their backend operations to large registry operators. For instance, Identity Digital's registry backend manages 447 new gTLDs, including `.group` and `.live` [26]. Thus, analyzing the public TLD zone files provides valuable insights into registry management practices.

## 2.2 EPP Operations and Domain Status Codes

Domain registries and registrars standardize domain registration and configuration processes according to the Extensible Provisioning Protocol (EPP) [27–29]. The protocol defines two primary types of resource objects: *domain objects*, which contain information about registered domain names, and *host objects*, which provide details about nameservers and delegation information. It ensures the secure isolation of domain name operations by allowing modifications to resource objects with proper authorization.

Registries operate all resource objects through EPP interfaces and standard EPP commands. Each registry customizes EPP interfaces for managing object repositories, enabling accredited registrars to create, update, and delete domain or delegation hosts. For example, when a registrant registers a DNS host object, the registrar first issues an EPP `<host check>` command to verify its eligibility for creation. If confirmed, an EPP `<host create>` command completes the registration. The DNS host object can then be modified or removed through EPP `<host update>` or `<host delete>` commands, respectively. However, varying EPP interface implementations require registrars to integrate multiple interfaces to manage resource objects across platforms.

EPP status codes, or domain name statuses, reflect the current phase in a domain's lifecycle. These codes show whether a domain is registered, revoked, or securely locked against unauthorized transfers, updates, or deletions. Status code transitions mirror changes in the domain lifecycle, with each stage potentially linked to specific codes. Additionally, certain status codes facilitate transitions through the domain lifecycle. For instance, the `ok` status signifies that a domain is in the `Registered` stage, while `pendingDelete` indicates the `Redemption Period`. However, codes like `serverHold`, which may be triggered by issues such as reported abuse, represent specific states rather than regular lifecycle stages.

## 2.3 EPP Operation Risks and Related Work

Improper EPP operations by registries or registrars can have profound effects on the domain name space. An existing work pointed out that the deployment of EPP servers could inflict catastrophic consequences on a TLD [50]. Akiwate et al. [16] brought to light a covert EPP operation that has inadvertently introduced a substantial volume of manipulable resource records within the domain name space. They uncovered a scenario where certain registrars have engaged in insecure renaming procedures on host objects to conform with the EPP deletion conditions. As for managing host objects in TLD zones, researchers have identified a substantial number of expired and exploitable glue records within the TLD's zone files [33, 48, 54].

# 3 Threats Across Domain Lifecycle

## 3.1 Domain Lifecycle: A Registry Perspective

A domain name progresses through various lifecycle stages formally defined by ICANN [10]. During these stages, registries manage domain and delegation host objects in TLD zones and update corresponding EPP statuses [11]. Common registry operations across the lifecycle are discussed in Appendix A. In this section, we delved into analyzing how the lifecycle stages correlate with specific EPP statuses, and broadly categorized the domain lifecycle into three phases: creation, management, and deletion, as shown in Figure 1. We detail these stages and their associated EPP statuses to provide a clear understanding of registry operations throughout

the domain lifecycle.

**Domain creation.** The creation of a domain name primarily involves two stages of the domain lifecycle: `Available` and `Add Grace Period`. It also encompasses three EPP statuses: `pendingCreate`, `serverHold`, and `addPeriod`. A domain name in the `Available` stage is not currently registered and is open for registration. When a registrant registers a domain name, `foo.example`, the registrar uses the EPP interfaces provided by the `.example` registry to create a domain object. This registration initiates the `Add Grace Period` stage, a specific timeframe (e.g., five days), during which registrants can cancel their registration requests. In this period, the registry conducts domain status and delegation checks. Concurrently, the necessary delegation hosts can be created upon registrants' settings. After completing all checks, the domain objects and the host objects are added to the TLD zone files.

**Domain management.** The domain management phase involves only the `Registered` stage, associated with various EPP operations. Routine operations involve statuses such as `inactivate` and `ok`. Additionally, registrants can initiate actions such as "renew", "transfer", and "update". These actions trigger the registries to execute corresponding EPP operations and modify the domains' EPP statuses.

In this phase, a registrant can configure three types of domain delegation hosts: in-domain delegation, sibling-domain delegation, and out-domain delegation [19]. The corresponding host objects are stored as NS or glue records in the TLD zone files. The EPP protocol requires that only pre-registered delegation hosts can be configured as NS records and stored in TLD zones. For example, to use `ns.foo.example` for `foo.example`, the registrant must first register it as a DNS host object (e.g., `ns.foo.example A 1.2.3.4`) and save it as a glue record. However, this restriction does not apply to out-domain delegation, as registries cannot verify the status of DNS hosts managed by other registries.

**Domain deletion.** Domain deletion is a complex process that involves multiple lifecycle stages and various EPP statuses. Registries must verify domain deletion requests through several stages, including the `Auto-Renew Grace Period` and `Redemption Period`, to prevent the accidental deletion of valid records. `pendingDelete` is the final status in the domain lifecycle; domains entering this stage are scheduled for removal after five days. At the domain lifecycle's end, registries must purge all associated resource objects, including domain and host objects.

Additionally, not all EPP statuses are associated with specific domain lifecycle stages or registrant actions. Registries can impose certain special EPP status codes at any point during the domain lifecycle under specific conditions. For instance, if a domain is involved in a legal dispute or is found to be abused, the registry may apply the `serverDeleteProhibited` or `serverHold` statuses, prohibiting resources from being deleted from the zone files and preventing relevant domain abuse.

## 3.2 Threat Analysis of Registry Operations

Registries face growing domain abuse threats while managing lifecycles. To address these issues, ICANN mandates measures for registries, including establishing a timely system to report and handle domain abuse [47]. However, the registries handle lifecycle operations and abuse prevention *internally*, leaving their security unanalyzed. In this study, we analyze the opaque registry operations from an *external* perspective using open datasets. Our analysis focuses on risks in three specific scenarios due to external constraints.

**Risks in domain creation.** Domain object creation is the starting point of a domain's lifecycle. Some registries and registrars implement pre-registration proactive measures (e.g., TrustName [22]) to defend against cybersquatting and domain abuses. For example, they set homograph domain names to a `Reserved` status and add them to zone files to guard against domain abuses. Our analysis revealed that certain proactive measures introduce additional creation for domain and host objects, thereby introducing new attack surfaces.

*Analysis idea.* TLD zone files contain all domain objects and DNS host objects (e.g., NS records) set by registries, while WHOIS servers maintain registration data and domain status. By combining these sources, we can track existing resource objects, compare them with WHOIS records to infer domain creation strategies (e.g., the creation of additional domain objects), and uncover potential security risks.

**Risks in domain management.** The DNS hierarchical structure and domain delegation dependencies make domain management complicated. Prior research showed that poor domain management can lead to problematic resource records, such as dangling records [40, 52], orphan DNS records [33, 48], stale glue records [54], and sacrificial NSes [16], which can appear in TLD nameservers. These mismanagement issues expose numerous domains to security risks. Despite this, a consensus-driven solution remains elusive due to unclear root causes, allowing these vulnerabilities to persist.

*Analysis idea.* Registries' management of DNS resource objects is reflected in daily updated TLD zone files, allowing for longitudinal analysis of record updates and horizontal comparison of practices across TLDs. By combining these analyses with proactive testing, we can externally assess how registries manage resource records and uncover the causes of problematic management practices.

**Risks in domain deletion.** Each domain is unique within the domain name space, registered for fixed periods (e.g., one or five years). Upon expiration, the registry deletes the domain name, making it available for new registration. During the deletion process, registries are required to remove all associated records from TLD zone files. However, complex delegation dependencies make this task challenging, leaving residual records in the zone files.

*Analysis idea.* Zone files reflect the presence of domains and delegation records, while domain registration information

(i.e., WHOIS) details domain statuses. The registry's operations facilitate the interaction between these two datasets, influencing each other. To this end, we can identify traces of registry deletion operations by conducting a comparative analysis of zone files and WHOIS data, exposing potential security risks in the deletion process.

## 4 Dataset and Analysis Methodology

We combine static and dynamic analysis methods to investigate the opaque operations of registries throughout the domain lifecycle. These methods rely on publicly available data and limited interactions with registrars via accessible interfaces to examine registry activities. This section describes the data sources, interfaces, and analysis methods.

### 4.1 Dataset

**TLD zone files.** A TLD zone file contains all delegation records, including NS and glue records, for SLDs within that TLD zone. ICANN CZDS [7] provides public access to zone files for generic Top-Level Domains (gTLDs) and updates them daily. The presence of a domain name in the TLD zone file indicates that recursive resolvers can obtain its resource records (e.g., SOA, NS, and glue records) from the TLD's authoritative nameservers, signifying its existence within the domain name space. We define this domain name status as *domain existence status*. In this research, we gather snapshots of the zone files for 1,109 gTLDs on June 25, 2024, which contain 219,468,943 domain names and 2,303,951 glue records.
**WHOIS records.** Domain registration details are recorded in publicly accessible WHOIS databases, which remain vital resources for DNS research [15, 54–56]. Though the General Data Protection Regulation (GDPR) requires WHOIS providers to redact registrant contact information, registration dates and other technical details (e.g., nameservers, registrar names) remain publicly available in WHOIS records. In addition, despite ICANN's plan to phase out WHOIS [44], registries and registrars continue to support it for compatibility. Thus, we primarily relied on WHOIS to analyze domain registration information in this study. We considered a domain registerable if its WHOIS status shows `available`, and we referred to this as the domain's *static registration status*.

However, collecting WHOIS for all domain names in our zone files is a difficult task. Registrars and registries do not allow users unlimited access to their WHOIS servers. Meanwhile, querying WHOIS records directly from WHOIS servers for all zone file domain names (~$10^8$) is time-consuming due to providers' rate limits (1k~5k daily queries). To address this, we prioritized historical WHOIS datasets and tried to supplement missing domains with active WHOIS queries. The WHOIS datasets include:

- *Historical WHOIS:* We accessed a passive WHOIS dataset collected by an industrial collaborating company. This dataset contains 7.9 billion historical WHOIS records (spanning January 2020 through July 2024) for 550 million effective SLDs, covering 89% of our zone file domains.
- *Active WHOIS:* We implemented a script using the `python-whois` package to actively crawl WHOIS records for the missing domains in the historical dataset. We executed the probing script on the collaborating company's distributed measurement platform, covering over 300 vantage points globally.

To sum up, the historical and active datasets cover 95.61% of the zone file domain names, providing a comprehensive basis for domain registration analysis. Due to ethical concerns, these WHOIS datasets are provided for research only, and all data analyses are conducted in a controlled environment authorized by the collaborator. The ethical consideration section provides more details.
**Registrar registration interfaces.** During the domain name registration process, registrars should verify the availability of the domain name using the EPP `<domain check>` operation in real-time. Upon receiving this request, registries evaluate whether the domain name is within its lifecycle and relay the status back to the registrar. This status decides if the registrant can proceed with registration. Thus, we define this promptly obtained domain status as *dynamic registration status*.

We collected the dynamic statuses of test domains using the interfaces provided by three registrars: DNSPod, Alibaba Cloud, and Dynadot. We opted not to use the APIs of well-known registrars like GoDaddy and Namecheap because they incorporate additional internal verification logic beyond simply verifying the domain registration status with the registry, which could skew our collection of the dynamic statuses.

### 4.2 Analysis Methodology

Our longitudinal analysis performs "static and dynamic resolution" of zone files and WHOIS data to identify potential issues. To illustrate our analysis, we use a mock zone file and WHOIS records (Table 1) to demonstrate how public data helps identify abnormal domain statuses.
**Static analysis.** First, we compare *domain existence status* in TLD zone files with *static registration status* in WHOIS records to identify vulnerable EPP operations during domain creation and deletion stages. For example, `bar.example` in Table 1 has NS records in the zone file, signifying its existence in the domain name space. However, its WHOIS information is currently unavailable, and historical WHOIS indicates that it expired on March 8, 2023. Thus, we infer that its *static registration status* is unregistered.

Second, we track the differences between the NS records in zone files and those in WHOIS to determine abnormal EPP operations in domain management. As shown in Table 1, `exa.example` is configured with two NS records in the zone

Table 1: Mock zone file and WHOIS records.

| Domain Name | Zone file | WHOIS[1] | | Registration Status | | Domain Status |
|---|---|---|---|---|---|---|
| | | NS | Expiry Date | Static | Dynamic | |
| foo.example | ns.foo.example | ns.foo.example | 2025-04-08 | registered | registered | normal domain |
| exa.example | ns.foo.example | ns.exa.example | 2025-04-08 | registered | registered | resurrected relic domain |
| exa.example | ns.exa.example | | | | | |
| bar.example | ns.bar.example | / [2] | 2023-03-08[3] | unregistered | unregistered | untapped relic domain |
| 测试.example (xn–0zwm56d.example) | ns.bar.example | ns.bar.example | 2025-04-08 | registered | registered | normal domain |
| 測試.example (xn–g6w251d.example) | ns.bar.example | / [2] | / [3] | unregistered | registered | twin domain |

[1]: We primarily focus on the NS records and the registry expiry date in WHOIS.
[2]: We are unable to obtain the WHOIS for that domain name from the registry/registrar's WHOIS servers.
[3]: We take the latest expiry date of the domain name in our historical WHOIS as the current expiry date. Therefore, although the domain name `bar.example` currently cannot request WHOIS, in the historical WHOIS its last expiry date was Mar 8, 2023. Moreover, "/" means that the domain name has never been registered before.

file, yet only one appears in the WHOIS record. Typically, each domain name registration should prompt the registrar to update the WHOIS database. Therefore, the presence of an additional NS record in the zone file could indicate a discrepancy due to flawed registry operations.

**Dynamic analysis.** Collecting *dynamic registration status* of domain names from TLD zone files enables us to identify irregular EPP operations. For example, 測試.example (xn-g6w251d.example) has NS records in the zone file but lacks accessible WHOIS records. However, the EPP <domain check> result indicates that it is registered.

We simulate registrant operations to explore registries' DNS host management models. Specifically, we purchase test domains across TLDs and registrars and then register different types of DNS host objects. We focus on two types of DNS host objects: those set as NS domains by other domains and those not associated with any domains. After registration, we check daily zone file updates to determine the TLDs' management strategies for different types of host objects. For example, with `foo.example`, we initially register two DNS host objects: `ns1.foo.example` and `ns2.foo.example`. We then configure only `ns1.foo.example` as the NS record. The next day, we check the zone file to see if both the two NS domains are present in the zone file. If `ns2.foo.example` also appears, it indicates that the `.example` registry unconditionally includes DNS hosts in the TLD nameservers even if they are not actively used.

# 5 Redundant Domain Creation

To begin with, we analyzed the pre-registration EPP operation performed by registries and registrars. During domain creation, while these entities take steps to protect trademarks and prevent domain abuse [22], some may create redundant domain objects, introducing new opportunities for domain abuse. In this section, we discuss the origin of this flaw and outline the threat model, followed by the detection method and an evaluation of the threat's real-world impact.

## 5.1 IDN Variants and Twin Domain Names

**IDN variants.** Redundant domain objects are often created when registering Internationalized Domain Names (IDNs) for certain regional languages. These IDNs may have linguistically equivalent character variants from different scripts, common in many language systems, such as simplified and traditional Chinese, Japanese Kana and Kanji. For example, a traditional Chinese IDN, 測試.example (xn-g6w251d.example), has a simplified Chinese equivalent, 测试.example (xn-0zwm56d.example). In the DNS community, IDNs with such equivalent character variants in labels are called *IDN variants*. They are interchangeable, sharing typographic, orthographic, and semantic similarities, and represent the same brand [31].

To support a multilingual Internet, ICANN and some registries have officially published policies for the registration and protection of IDN variants, including maintaining ownership and nameserver settings for the IDN variants of registered IDNs [12,38]. Most gTLD and ccTLD registries also maintain IDN Tables that list the equivalent character variant ranges to standardize the IDN registration [5, 13, 14].

**Twin domain names.** Our analysis uncovered that domain registries auto-create one or more IDN variants for trademark protection and abuse prevention during IDN creation, which we refer to as *twin domain names*[1]. As shown in Figure 2, when a user aims to register 测试.example, the registry of .example will automatically and simultaneously create its IDN variant, 測試.example. This practice has been widely

[1]While there may be more than two equivalent IDN variants, we collectively refer to them as twin domains for simplicity.

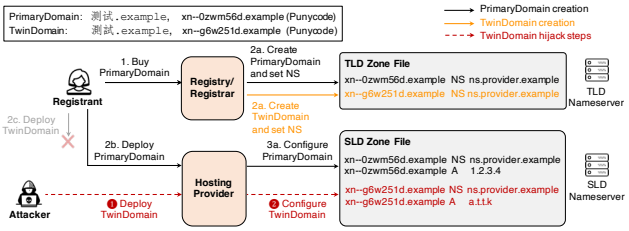


Figure 2: Twin domain creation and attack steps. Note that steps 2a and 2b involve multiple interactions, so their order is not strictly fixed. The figure provides a simplified illustration.

adopted by many registries to reduce the potential confusion in domain creation.

**Twin domain names v.s. Homograph domain names.** Prior work proposed IDN homograph attacks, where attackers register visually similar domains to spoof users [30,39,49]. However, twin domains differ from homograph domains in two key aspects. First, twin domains are IDN variants representing the same entity or brand, whereas homograph domains are visually similar but differ in semantics, orthographics, and ownership. Second, twin domains are created by domain registries, while homograph domains are registered by attackers.

## 5.2 Threat Model

**Attack surface.** We uncovered EPP implementation flaws that pose domain abuse risks when registries create IDN domain objects. As shown in Figure 2, when a victim registrant attempts to buy an IDN (*PrimaryDomain*), the registry simultaneously auto-creates domain objects for its IDN variants (*TwinDomains*) per IDN Tables and sets identical NS domains (e.g., ns.provider.example) without notifying the victim registrant. Namely, the registrant-configured *PrimaryDomain* and registry-configured *TwinDomains* are delegated to the same SLD NSes, which are typically managed by the DNS services of registrars (e.g., Godaddy) or third-party hosting providers (e.g., Cloudflare). However, the SLD NSes only provide authorized DNS records for the *PrimaryDomain* after domain registration. Hence, if a client queries *TwinDomains*, the SLD NSes will answer with `REFUSED` or `NXDOMAIN`.

In this scenario, we consider that the *TwinDomains* could be vulnerable if their SLD NSes belong to public hosting providers (e.g., CDNs, DNS hosting, web hosting), which provide no authorized DNS records for the *TwinDomains*. Previous work has identified that dozens of public hosting providers (e.g., Amazon Route 53, GoDaddy DNS, Alibaba Cloud DNS) perform weak domain validation during customer domain deployment or reuse a small nameserver pool to allocate NS domains for customers [18, 23, 32, 52, 56]. These providers allow customers to deploy any domain names without authority. In this circumstance, attackers can create accounts from such a provider, apply the same nameservers (e.g., ns.provider.example) as the victim registrants, and deploy *TwinDomains* on the platform (Step ❶ in Figure 2). They then configure malicious DNS records via the nameservers to hijack the *TwinDomains* (Step ❷). Note that public DNS hosting services, used by 89% of the top 100K websites [34,43], allow customers to unauthorized claim any domains, even if the domains do not belong to them [18,51,56]. This makes leveraging hosting services with flawed domain deployment processes to hijack *TwinDomains* a practical attack vector.

**Twin domain takeover threat model.** Assume the NSes (e.g., ns.provider.example) of *TwinDomains* and *PrimaryDomain* belong to a provider with weak domain ownership validation during domain deployment processes. Attackers can be any malicious customers of the provider. Their ability only includes: 1) registering multiple free accounts from the provider; 2) repeatedly applying for new NS domains to match *TwinDomains*' current NSes, and 3) claiming the ownership of arbitrary domain names without authority and configuring DNS records for them. With these abilities, they can set malicious records for *TwinDomains* via the interfaces or console webpages provided by the provider. Their attack goal is leveraging the hosting provider to covertly hijack registrant-unknown *TwinDomains* and abuse them for illegal activities.

**Attack steps.** To start a twin domain takeover attack, an attacker must: 1) identify a *TwinDomain* whose NSes belong to a hosting provider that does not perform domain ownership verification; 2) check *TwinDomain*'s DNS resolution status (e.g., `NXDOMAIN`, `REFUSED`) to ensure the provider's NSes are not configured with authorized DNS records for the *TwinDomain*; 3) apply the same NSes using the attacker accounts and deploy *TwinDomain* on the platform; 4) configure malicious DNS records (e.g., `A` records) to the zone files of the NSes, redirecting *TwinDomain* to an attacker-controlled IP (e.g., a.t.t.k). Then, all client's DNS queries for the *TwinDomain* will be recursively led to the attacker's IP. In this case, the attacker can abuse the *TwinDomain* for malicious purposes.

**Threat implications.** Twin domains are created by registries and registrars, with nameservers configured by these entities. Although the twin domains are resolvable via TLD nameservers, their registrants (i.e., the theoretical owners) are typically unaware of their existence and cannot access or manage them through their registrar accounts. Thus, they are unable to remove or modify the NS records associated with these domains. Attackers can exploit TLD zone files to identify such twin domains, and abuse them following the aforementioned attack steps for illegal activities, like distributing malware and establishing malicious websites, as illustrated in Appendix B. Moreover, since the twin domains in the same equivalent group are interchangeable, threats persist regardless of which domain is registered first (i.e., becomes the *PrimaryDomain*).

## 5.3 Identifying Twin Domain Names

During domain registration, registries create domain objects and set NS domains as per the EPP standard [28]. Since

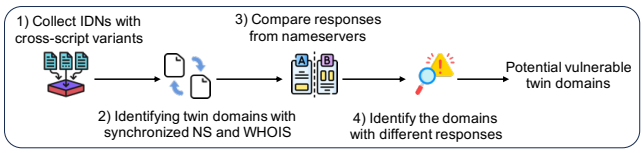


Figure 3: Workflow of identifying vulnerable twin domains.

twin domains are created and configured simultaneously by registries, they share identical creation dates with domain objects and parallel NS domains in host objects. Based on this, we design a workflow to identify twin domains and detect vulnerable ones, as shown in Figure 3.

**Grouping IDN variants in TLD zone files.** We begin by extracting all IDNs from TLD zone files and identifying linguistic-equivalent IDN variant groups using language-conversion libraries, such as `zhconv` (v1.0.3) and `greek-language-tools` (v1.0). For example, IDNs with equivalent traditional and simplified Chinese characters are grouped together. The IDN variant groups containing more than one IDN are selected for further analysis.

**Identifying twin domain name candidates from IDN variants.** We assume that twin domain names configured by registries share *identical creation settings* (e.g., creation dates and registrar organizations) in WHOIS and *identical NS settings* in TLD zones. To this end, we examine the registration information of IDN variant groups using the WHOIS dataset and the TLD zone file dataset described in Section 4.1. For each IDN variant group, we compare the `CreationDate`, `NameServer`, `Registrar`, and `WhoisServer` fields of the WHOIS records, as well as the IDNs' `NS records` in the zone files. To account for slight variations in the exact creation time at daily granularity, we extract only the date parts from the `CreationDate`.

**Detecting vulnerable twin domain names.** Our threat model assumes that registries create twin domains and assign NS domains (i.e., SLD nameservers), allowing their NS records to be resolved via TLD nameservers or extracted from TLD zone files. Meanwhile, registrants are unaware of the twin domains' creation and do not configure any resource records, so the SLD nameservers will not provide authoritative resolution responses. This results in inconsistencies between the DNS responses from the parent zone (TLD nameservers) and the child zone (SLD nameservers), leaving the twin domains vulnerable to takeover threats.

Specifically, when a client queries the NS records for a twin domain, the TLD nameservers can successfully answer the registry-maintained zone data, while the SLD nameservers return `REFUSED`, `NXDOMAIN`, or other response codes (rcodes) because they do not maintain any authorized DNS records for the twin domain. Thus, the detection method involves comparing the DNS resolution responses between the TLD and SLD nameservers within the twin domain groups. Possible threat scenarios can be categorized by different rcodes:

- *S1: SLD nameservers respond with REFUSED (rcode=5).* Some hosting providers (e.g., Alibaba Cloud DNS) refuse to answer any queries for domain names that are not deployed on their platform. In this scenario, an attacker can take over the twin domains by deploying them on the vulnerable hosting providers and applying the same NS domains as Section 5.2 introduced.
- *S2: SLD nameservers respond with NXDOMAIN (rcode=3).* Some DNS providers respond that the queried twin domains do not exist in the zone. In this case, the attacker can also launch domain takeover attacks, as in scenario S1.
- *S3: SLD nameservers respond with NOERROR (rcode=0).* If the providers successfully answer the queries for twin domains but provide different answers from the primary domain, further investigation for HTTP response contents (e.g., HTML documents) of the twin domains is required to assess potential domain abuse.

## 5.4 Evaluating Twin Domain Threats

In this section, we first present the inferred twin domain creation patterns based on public TLD zone file data. This analysis offers a coarse-grained perspective on the current twin domain landscape from a registry-level viewpoint. However, since zone files only capture the final state of domain management by registries and registrars, the results should not be interpreted as direct evidence of registry practices. Next, to identify twin domains that are truly exposed to takeover risks, we conducted active DNS resolution checks and report the confirmed set of vulnerable domains.

**Exploration of twin domain creation patterns under various registries.** We inspected the domain creation phase by registering test IDNs, analyzing collected TLD zone data, and confirming twin domains of their registered IDNs. Eventually, we uncovered different practices in handling IDN variants employed by domain registries. We identified twin domains in 61 TLDs. The top three TLDs by count of these domains were `.top` (17,615), `.商城` (14,444), and `.我爱你` (3,427), accounting for over 72.6% of the total.

Our analysis showed that twin domain group sizes vary with character types due to differing numbers of equivalent characters, even within the same registry and policy. For example, a CJK character U+3447 has 4 variants, while U+4E7E has 8 [5]. Despite this variation, we found that registries of Chinese IDNs create a higher number of variants, likely due to their official policies aimed at protecting brand and trademark rights for both simplified and traditional Chinese equivalents [9]. Among them, the `.cn` TLD has published an IDN table for Chinese IDNs and included variants in the zone [37].

In addition, our experiment results indicate that CNNIC groups all twin domains into a single WHOIS record and employs parallel NSes for the variants. However, despite the NS configuration from the parent zone, the SLD nameservers

might return REFUSED or NXDOMAIN if the twin domains do not initialize hosting services on the DNS platform as scenario *S1* and *S2* in Section 5.3. In this situation, attackers could abuse the twin domains without paying fees by allocating shared SLD nameservers from the hosting platforms. Similarly, TWNIC and Telnames automatically register up to two additional Chinese IDN variants.

Most registries do not bundle or block IDN variants for other languages or scripts, with the exceptions being the .cat TLD, which bundles Catalan IDNs with their ASCII equivalents, and the .gr TLD, which bundles punctuated and unpunctuated forms of Greek IDNs. Though these registries claim to register IDN variants, we did not identify relevant cases from our data.

**Identified twin domains and confirmed vulnerable ones**. From the zone files, we extracted 1,290,170 punycode IDNs (starting with the "xn--" prefix), which contain 462,057 IDNs with domain variants. After mapping linguistic-equivalent script forms, we identified 46,135 potential IDN variant groups, encompassing 92,348 IDNs. Additionally, we gathered 8,887 more domain candidates from the domains project on GitHub [20], resulting in 4,387 groups. Altogether, we collected 467,251 IDNs with character variants, forming 48,871 groups and covering 97,542 IDNs. Most (99.7%) of these groups contain only two IDNs, while the rest have up to five IDN members. Further, we discovered 8,866 IDN groups with synchronized registration information, making them potential targets for twin domain takeover attacks.

Finally, we confirmed vulnerable domains by querying NS records from both TLD and SLD nameservers. Among the 97,542 IDNs in twin domain groups, we identified 70,242 (72.01%) exhibited domain resolution inconsistencies and 6,017 (6.17%) vulnerable twin domains, revealing all of the three scenarios (*S1*-*S3*):

- For *S1*, we uncovered 3,582 vulnerable twin domains that have been delegated to hosting providers exploitable for domain takeover threats. The providers include Alibaba, Baidu, Bizcn, and Bigwww.
- For *S2*, we discovered 1,186 vulnerable twin domains with NS domains configured to DNSPod, a prominent DNS service by Tencent Cloud.
- For *S3*, we identified 28,311 twin domains with resolution inconsistencies, including 3,441 IDNs across different script variants. Further, we examined the HTTP responses of these twin domains and found that 1,249 of them were fraudulent. Specifically, we confirmed that they were abused and had deployed sophisticated cloaking technologies [53], including delayed display and access restrictions (e.g., upon multiple visits to a domain, it redirects to Google). Appendix B presents the cases of twin domains identified as being abused.

### 5.5 Discussion

**Limitation.** First, our analysis method only supports cases where both the user-registered domain and its twin domains are present in the TLD zone files. We are unable to capture cases where the original domain has expired but the twin domain remains.

Second, our twin domain identification process may introduce false positives due to the coarse date-based matching approach. For instance, two different registrants or registrars might coincidentally register similar domains on the same day. However, such coincidences are likely less common than instances driven by enforced IDN variant policies from registries. Moreover, our method serves as a coarse-grained prefilter to identify potential twin domain candidates. To assess actual vulnerability and threat implications, we further verify the DNS status of each candidate and confirm exploitability through follow-up procedures.

While these limitations constrain a complete view of the twin domain takeover threat, the dataset still effectively highlights the issue. Achieving more comprehensive and accurate detection would require real-time monitoring of domain status and zone file updates.

**Root cause and mitigation strategies.** Twin domain takeover threats arise when registries create twin domain objects for brand protection during IDN registration and assign the same nameservers as the primary IDN without informing registrants. However, these nameservers may not be fully controlled by registrants. If they are managed by public providers and shared across customers, attackers can apply the same nameservers to manipulate the registrant-unknown twin domains. To prevent potential domain abuse, we suggest registries notifying registrants when creating twin domains, and configuring them to registry-controlled nameservers or not assigning nameservers as default. Some registries, like CNNIC, have confirmed this approach.

## 6 Siloed DNS Host Management

Existing research shows that TLD zone files contain stale glue records [15, 54]. Although these glue records are still referenced by other domain names, they cannot provide effective authoritative services and may even provide attackers with hidden attack vectors, which highlights the need for host management guidelines. However, the cause of host mismanagement remains unclear, with registry managers often unsure which host objects cause issues. This leads to ad-hoc object removals, sometimes mistakenly deleting active glue records, which can compromise domain resolution integrity.

In our analysis, we identified the causes of stale glue records during the domain management phase. We revealed the improper EPP operations, specifically that some registries indiscriminately add all DNS host objects to zone files, contributing to their proliferation. In this section, we examine

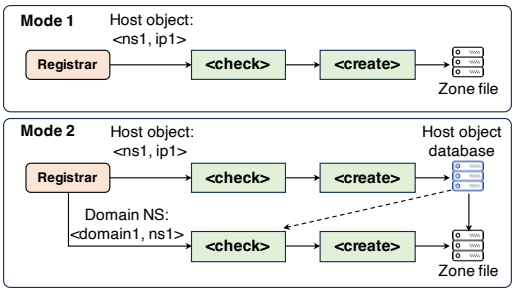


Figure 4: Two different DNS host management modes.

registries' DNS host management modes, identify flawed practices, and clarify the threat model and its impact.

## 6.1 DNS Host Management Mode

**Exploration of host object management modes.** We leveraged the dynamic analysis method in Section 4.2 to investigate DNS host management practices across registries. Specifically, we selected the top 10 TLDs by domain registration volume, including `.com`, `.net`, and `.org`, covering over 90% of registered domains for testing. We also selected four leading registrars—GoDaddy, Namecheap, Alibaba Cloud, and Tencent Cloud—that support DNS host configuration, each registering over a million domain names. Then, we registered 20 domain names from each registrar under different TLDs and conducted DNS host configuration tests. After checking the NS updates in zone files, we performed host deletion tests to assess EPP compliance, which prohibits deleting DNS hosts if they are relied upon by any domain objects.

**Identified host management modes.** Our experiments found that some registries employ conservative strategies in managing host objects; specifically, these objects are added into zone files even in the absence of in-domain delegation. In a typical scenario, when a user submits a DNS host (e.g., glue record) registration request, the registrar initiates an EPP `<host check>` command to verify the registration eligibility, such as checking user permissions and existing registrations. Upon a positive response, the registrar proceeds with an EPP `<host create>` operation to register the DNS host. Based on the dynamic analysis and public TLD zone files, we abstracted two high-level patterns that may be used by domain registries to manage DNS host objects, as illustrated in Figure 4:

*Mode 1*: The registry accepts the queries, creates the DNS host object, and adds it to the TLD zone file without checking for in-domain delegation. We refer to these as *M1 glue records* and demonstrate that they can be exploited by adversaries to hijack or block access to benign domain names.

*Mode 2*: After accepting the queries, the registry saves the DNS host to a separately maintained database containing all submitted host objects. It only adds the host object to the zone file when it is used in an in-domain delegation, avoiding direct and unconditional object inscription.

The two DNS host management modes are empirical abstractions based on observed patterns in public data, rather than exact representations of registry or registrar workflows. Real-world operations are often more complex, and our models capture only high-level behaviors. Based on the abstraction of these two operational modes, our experimental results demonstrate that the differences between these modes lead to significant variations in stale glue records within zone files.

## 6.2 Threat Model

**DNS host management threats.** By dynamically tracking DNS host updates in zone files, we observed that some host objects under major TLDs like `.com`, `.org`, and `.net` follow Mode 1, which appears to be a key cause of the stale glue records. The DNS hosts following this mode are added to zone files regardless of their usage, neglecting their intended role in domain delegation. If registrants forget to remove these hosts after updating NS configurations, stale glue records can accumulate in the zone files.

The abuse of stale glue records is a pervasive problem, as noted in previous studies [15, 54]. In this threat scenario, attackers can be any malicious actors who can access public TLD zone files [7] to identify stale glue records that are relied on by victim domains. The attackers aim to exploit the stale glue records to take over or block access to benign domain names. They can target legitimate users by injecting the stale glue records into specific DNS resolvers, which requires public DNS resolvers to cache and utilize stale glue records[2].

Assuming that the DNS resolvers rely solely on the authoritative nameserver specified in the stale glue record, even if they have stopped responding. In this case, the attackers can exploit these stale glue records to carry out denial-of-service (DoS) attacks on the target domains or answer with fake resource records by obtaining the NS IP addresses. Mainstream resolver implementations (e.g., BIND9 and PowerDNS) have been demonstrated to meet the aforementioned requirements [54].

**Attack steps.** Attackers exploit stale glue records from Mode 1 operations through the following steps: 1) Identify exploitable DNS host objects through public TLD zone files. 2) Inject them into the target DNS resolver. 3) Acquire the IP addresses of the stale glue records by continuously allocating and releasing IPs via cloud service providers. 4) If successful, deploy rogue authoritative nameservers to serve fake resource records. Otherwise, exploit the resolver's trust in stale glue records to force domain resolution failures.

**Threat implications.** The impact of mismanaged hosts in Mode 1 is twofold. First, registries are required to maintain a large number of stale glue records, which cannot be easily cleaned up due to their dependence on numerous domains.

[2]To cache a record in a resolver, the attackers should configure specific resource records for a domain they control and initiate queries to the target resolver [54]. Appendix C presents a sample of the attacker's resource record configuration.

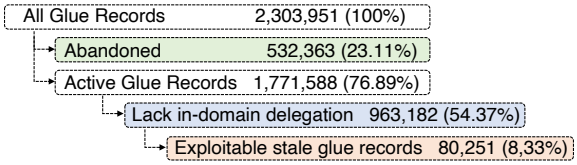


Figure 5: Statistics of glue records.

Second, these stale records directly affect user access to active domain names. When legitimate users visit a domain that relies on stale glue records, they could receive fake resource records from attacker-controlled authoritative nameservers. These records may redirect them to malicious websites, such as phishing sites or malware distribution platforms. They may also experience service disruptions when accessing the domain. The impact of the DoS attack depends on the stale glue record's cache duration, which typically lasts up to one hour per attempt. Furthermore, since attackers simply reactivate dormant stale glue records in the domain resolution chain, the attack is highly covert.

## 6.3 Evaluating Impact of Mode 1

**Identifying glue records.** We used the zone file dataset to assess Mode 1's impact on managing TLD resource records. Specifically, we extracted the complete set of glue records from the zone files. Then, we conducted a refined sifting process to discover the glue records that lack in-domain delegation. Moreover, we replicated the methodology from [54] to identify exploitable stale glue records.

**Identified exploitable glue records matching the Mode 1 pattern.** Figure 5 illustrates the statistics of glue records. We extracted 2,303,951 glue records from the zone files of 1,109 TLDs. Of these, 532,363 (23.11%) were identified as abandoned (i.e., not associated with any domain names), while 1,771,588 are active glue records (i.e., having delegation relationships). Among the active glue records, we found that 963,182 (54.37%) lacked in-domain delegation. Of these, 430,820 exhibited sibling-domain or out-domain delegation. Additionally, we identified 184,308 exploitable stale glue records serving 6,068,353 domain names. Of these, 80,251 (43.54%) lacked in-domain delegation, affecting 1,600,253 (26.37%) domain names. Table 2 lists statistics of glue records associated with Mode 1 across the top 10 TLDs. The `.com` zone file contains 356,762 M1 glue records that lack in-domain delegation, with 47,593 exploitable for attackers. Common TLDs such as `.org`, `.info`, and `.net` also have a significant number of M1 glue records.

## 6.4 Discussion

**Comparison with relevant work.** Zhang et al. [54] highlighted the threats posed by stale glue records and assessed their real-world impact on registries and DNS providers, urging registries to remove stale records. However, it remains unclear why these stale glue records persist in the zone files maintained by registries for such extended periods. To address this gap and fundamentally prevent the creation of new stale glue records, our work examines DNS host object management policies employed by TLD registries from the perspective of EPP protocol implementation. We identify improper host object management as a key factor contributing to the persistence of stale glue records. Our work is not intended to directly clean up existing stale glue records. We aim to help registries understand existing shortcomings and guide them in correcting vulnerable practices to avoid introducing new stale glue records.

Table 2: Top 10 TLDs with stale glue records following Mode 1 pattern.

| TLD | # Glue record[1] | # Exploitable[2] | Percentage |
|---|---|---|---|
| .com | 356,762 | 47,593 | 13.34% |
| .org | 287,758 | 18,926 | 6.58% |
| .info | 96,668 | 4,157 | 4.31% |
| .wtf | 83,479 | 19 | 0.02% |
| .net | 47,262 | 3,707 | 7.84% |
| .top | 13,581 | 1,645 | 12.11% |
| .live | 8,525 | 384 | 4.50% |
| .pro | 7,838 | 324 | 4.13% |
| .life | 5,231 | 915 | 17.49% |
| .digital | 3,533 | 86 | 2.43% |

[1]: Glue records lacking in-domain delegation.
[2]: Exploitable stale glue records.

**Root cause and mitigation strategies.** Managing DNS host objects in Mode 1 does not maintain additional databases, which only requires a simple implementation without complex checking. Thus, this practice is common among traditional gTLDs like `.com` and `.org`. While not violating RFC specifications, it presents challenges for managing TLD authoritative servers. To mitigate these risks, we propose the following recommendations: First, registries using Mode 1 should transition to Mode 2 to prevent the accumulation of stale glue records. Second, for existing stale glue records, registries should enhance their current stale resource record cleanup mechanisms. For example, they could monitor the status of delegation records to promptly identify stale glue records. Additionally, since stale glue records are still being used by other domains, they cannot be deleted outright. Instead, they should be frozen for a certain period before deletion to ensure that legitimate domains are not affected. Third, registrants should monitor the status of their authoritative nameservers and promptly clean up stale glue records. And they should also actively implement domain security mechanisms, such as DNSSEC. Finally, registries, registrars, and registrants could establish efficient feedback mechanisms to address potentially flawed resource records in a timely manner. For instance, registries can provide feedback to reg-

istrants regarding configuration issues with their authoritative name servers through registrars.

## 7 Vulnerable Domain Deletion

Our analysis of domain deletion operations revealed vulnerabilities in EPP implementations that can falsely terminate a domain's lifecycle. As a result, even expired or unregistered domain names remain vulnerable to exploitation. Combining zone files and WHOIS datasets, this section further analyzes the issue and highlights its prevalence.

### 7.1 Relic Domain Names

**Relic domains.** According to the EPP specification, registries should remove all associated records during the domain deletion phase, including DNS host objects and WHOIS records. Thus, the *domain existence status* (whether delegation records exist in zone files), the *static registration status* (whether WHOIS records are present), and the *dynamic registration status* (obtained through the EPP `<domain check>` operation) should be reset, allowing the domain to enter a new lifecycle. However, we observed that registries' fragile deletion implementation prevents the *domain existence status* from being reset. Specifically, some domains' static and dynamic registration statuses are reset, making them appear unregistered from the view of registries and registrars. Yet, their delegation records have not been deleted from the zone file. As a result, the residual delegation records from the previous lifecycle remain permanently in place, and we refer to such domains as *relic domain names*.

**Categories of relic domains.** We categorized relic domains into two types based on whether they have been re-registered:

*Resurrected relic domains.* These relic domains have been re-registered by new registrants, entering a new domain lifecycle. However, since the current registrar lacks management rights over the residual delegation records, they can add new records but cannot remove the old ones. As a result, the WHOIS data reflects the updated NS records, while the zone file still retains outdated NS records, causing inconsistencies between the two. As shown in Table 1 for the domain `exa.example`, the zone file contains two distinct NS records: `ns.foo.example` and `ns.exa.example`. However, the WHOIS record only lists `ns.exa.example`. This discrepancy suggests that `exa.example` is a resurrected relic domain, with `ns.foo.example` being a leftover delegation record from the previous lifecycle.

*Untapped relic domains.* These relic domains have not been re-registered, meaning they lack new WHOIS records. For example, as shown in Table 1, `bar.example` has a delegation record, `ns.bar.example`, in the zone file but no current WHOIS record. The historical WHOIS data indicates it expired on March 8, 2023.

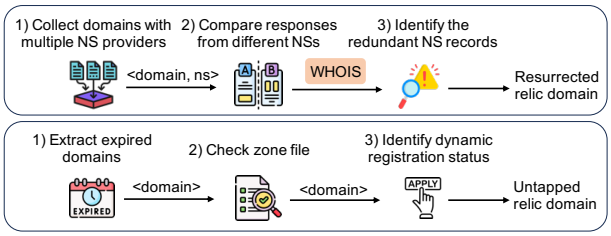


Figure 6: Workflow of identifying relic domains.

### 7.2 Threat Model

**Relic domain threats.** We uncovered that relic domains could also pose domain abuse risks. In this threat model, attackers attempt to acquire some relic domains at no cost and abuse them to conceal malicious activities. To exploit these relic domains, attackers need 1) access to publicly available TLD zone files and domain registration data, and 2) the ability to apply the same nameservers for relic domains with public DNS hosting providers, similar to the twin domain takeover model (in Section 5.2). The attackers could exploit the residual delegation records of the relic domains to take over these seemingly unregistered domains and manipulate their resource records to carry out malicious activities.

**Attack steps.** Attackers first analyze TLD zone files and domain registration data to identify relic domains with residual NS records. They then filter for those with NS records pointing to vulnerable hosting providers with domain validation flaws. Finally, they deploy the relic domain on the hosting providers and take control of them, by allocating the same NS domains, enabling unauthorized use and abuse.

**Threat implications.** This threat poses a significant disruption to the DNS ecosystem. On the one hand, it disrupts the traditional domain registration process, allowing attackers to control unregistered domains for illicit activities freely. This not only reduces the revenue of registries but also bypasses the abuse monitoring typically enforced by registries and registrars for newly registered domains. On the other hand, attackers' control over relic domains is nearly permanent before our work is disclosed. This is reflected in two aspects: 1) relic domains cannot be identified by the regular invalid record cleanup policies of registries, allowing them to persist indefinitely within the domain name space; 2) the lack of registration attributes further impedes analysis by security researchers. Moreover, the residual delegation records will not be removed, even if the domains are re-registered. In this scenario, current domain registrants are unaware of the existence of relic domains, and if attackers continue to hijack them for illicit activities, it will harm their reputation.

### 7.3 Identifying Relic Domains

As defined in Section 7.1, the key characteristic of relic domains is the NS inconsistency between zone files and WHOIS records. Resurrected relic domains have different NS records

in the zone files and WHOIS, while untapped relic domains have NS records in the zone files but lack accessible and up-to-date WHOIS data. Thus, an intuitive approach to identifying relic domains is to extract all domain names from the zone files, collect and compare their NS records in WHOIS, and check if they are available for registration.

However, implementing it is challenging. First, although we have collected a large-scale WHOIS dataset, it does not contain all updates made by registrants. Second, domain delegation records may change during the collection process, leading to false positives when directly comparing zone files and WHOIS records. To address these challenges, we designed an identification approach leveraging the inherent characteristics of the two types of relic domains, as shown in Figure 6.

**Identifying resurrected relic domains.** Directly comparing the NS records of all domains in the zone files and WHOIS is costly and ineffective in accurately identifying resurrected relic domains. Instead, we examine response differences across various NS domains to identify potential resurrected relic domains. Resurrected relic domains, which have both NS records from previous lifecycles and newly added ones, will exhibit different responses because the residual nameservers have stopped service for them.

Specifically, we first extract domain names that rely on multiple NS service providers from TLD zone files. Using methods from previous work [56], we identify service providers based on their NS domains' SLDs. For instance, domaincontrol.com and cloudflare.com correspond to GoDaddy and Cloudflare, respectively. While a single NS service provider may offer multiple distinct NS SLDs, this only broadens the potential set of resurrected relic domains without affecting the results. In addition, we group the NS SLDs together that contain domain labels for service providers with distinct identifiers (e.g., Amazon Route 53). Then, we select one NS from each group for relevant relic domain names to construct <domain, ns> pairs. Afterward, we query the domain name to the NS in each pair. If we receive different responses for a domain name across various nameservers, we consider it a potential resurrected relic domain.

To account for load balancing, some domain names may have different resource records across nameservers. These domains are also flagged as potential resurrected relic domains, but will be filtered out in the next step. Subsequently, we extract the WHOIS records for the potential resurrected relic domains from our WHOIS dataset to check the NS differences, i.e., some NS records in zone files are absent in WHOIS. Finally, we employ the static analysis method described in Section 4.2 to confirm the resurrected relic domains. Note that we manually verify each resurrected relic domain to exclude false positives.

**Identifying untapped relic domains.** We utilize the static analysis method introduced in Section 4.2 to identify potential untapped relic domains, and then confirm the domain's *dynamic registration status* using dynamic analysis methods to recognize untapped relic domains. Specifically, we first extract domain names from the WHOIS dataset with an `unregistered` static registration status, indicating that the domains have expired. Using a zone file snapshot, we then verify their *domain existence status*, i.e., whether they are present in the zone file. For domain names with an `unregistered` static registration status but a `valid` domain existence status, we further collect their *dynamic registration status*. This is done through a registrar we collaborate with, as we are not registrars and cannot directly perform EPP operations with the registry. Finally, if a domain's dynamic registration status is `unregistered`, we label it as an untapped relic domain.

## 7.4 Evaluating Impact of Relic Domains

**Resurrected relic domains.** We identified 19 resurrected relic domains involving 13 TLDs, which include one gTLD (`.biz`) and 12 new gTLDs (e.g., `.top`, `.zone`, and `.wiki`). Table 3 lists the statistical information of the new gTLDs affected by relic domains and their registry backends. Our new gTLD registry backend data is derived from a public project, nTLDStats [26]. It provides domain name data covering 1,114 new gTLDs, including statistical information on registrars, registries, and registry backends. The 12 affected new gTLDs rely on four registry backends: Identity Digital, GoDaddy, Nominet, and ZDNS. Collectively, these four registry backends support 756 new gTLDs.

**Untapped relic domains.** We discovered 3,425 untapped relic domain names involving 11 new gTLDs, including `.top`, `.app`, and `.cymru`. These new gTLDs are deployed across five registry backends, including GoDaddy, Nominet, Google, ZDNS, and Beijing Tele-info Network Technology, as shown in Table 3. Moreover, we found that new gTLDs supported by the ZDNS registry backend dominate the discovered untapped relic domains, accounting for 98.5% of the total. Therefore, we engaged in discussions with ZDNS regarding the specifics of the issue. Eventually, we confirmed that the problem stemmed from an abnormal EPP status within the domain name lifecycle. ZDNS has implemented mitigation strategies and fixed the affected domains and new gTLDs. The detailed discussion is outlined in Section 7.5.

Additionally, we inspected the NS records of these relic domains, primarily associated with registrars and hosting providers, and confirmed they are vulnerable to exploitation. Specifically, attackers could hijack these domains through unauthorized claims from DNS hosting service providers [18]. Using the method outlined in [56], attackers could also take over relic domains delegated to the registrar's nameservers.

## 7.5 Discussion

**Root causes.** Relic domains have had a widespread impact within the domain name space. However, it is difficult to ascertain their root cause based solely on registration data and

Table 3: Affected new gTLDs and registry backends of the relic domain.

| Regstry[1] | #TLD[2] | Resurrected relic domain | | Untapped relic domain | |
|---|---|---|---|---|---|
| | | TLD | #Domain | TLD | #Domain |
| Identity Digital [3] | 447 | zone | 1 | - | - |
| GoDaddy [24] | 216 | courses, design, party, club, wiki, rugby | 8 | vip | 1 |
| Nominet [45] | 73 | cymru, bot, wales | 3 | cymru | 4 |
| Google [25] | 46 | - | - | app, page, dev, みんな | 47 |
| ZDNS [8] | 20 | top, ren | 5 | top, wang, ren, 我爱你 | 3,374 |
| Beijing Tele-info [6] | 10 | - | - | 信息 | 1 |
| **Total** | **812** | - | **17** | - | **3,425** |

[1]: The relationship between new gTLDs and their registry backends is derived from the project nTLDStats [26].
[2]: The number of new gTLDs supported by the registry backend.

zone files. Therefore, we collaborated with ZDNS, the registry backend with the highest number of relic domains, to conduct an in-depth analysis of the causes of relic domains. Through case studies combined with source code review, we determined the cause of relic domains generated by ZDNS. During the final stage of the domain name lifecycle, `PendingDelete`, the abnormal superposition of EPP status codes disrupted the normal processing logic, resulting in the domain name object not being removed from the domain name space after the termination of the domain name lifecycle. As introduced in Section 3.1, at the end of the domain name lifecycle, the registry must clear all related domain resource objects, including registration data, DNS host objects, domain name objects, etc. However, when a domain is in the `PendingDelete` status and simultaneously in the `serverHold` status, the overlap of these two statuses causes the domain lifecycle processing logic to fail. As a result, the registry backend only clears the domain registration data and DNS host object data.

**Mitigation strategies.** To mitigate the threat, we propose the following mitigation measures for registries: 1) Timely repair of defects. After the discussion, ZDNS and Nomulus quickly fixed their defects and addressed the existing relic domains. 2) Targeted detection and assessment. Registries can implement domain lifecycle health detection methods as we introduced in Section 7.3 by combining data from authoritative servers with domain registration data provided by registrars to timely identify potential defects. We are also collaborating with registries to plan the launch of domain lifecycle anomaly assessment methods.

## 8 Disclosure and Limitations

**Disclosure.** We responsibly disclosed the identified issues to the relevant registries and their backend service providers. For the twin domains, we have reported to the registries (such as CNNIC, ZDNS, Nomulus, etc.) and discussed mitigation strategies with them. We also reported to Verisign with our findings. Finally, we engaged in detailed discussions with ZDNS and Nomulus regarding relic domains, assisted in implementing fixes, and ensured the issues were promptly resolved. We plan to contact more registries and affected registry backends through the ICANN DNS community for better mitigation.

**Limitations.** Despite using multiple data sources to ensure comprehensiveness, our study still has some limitations:

First, we have not analyzed domain names under country-code top-level domains (ccTLDs) due to the difficulty of obtaining data, despite their potential for using a broader character set. We mainly focus on new gTLDs and the most prominent open-source `tb0hdan/domains` dataset on GitHub. We believe this dataset can provide practical insights into the issue. In the future, we will strive to extend our analysis to include ccTLDs.

Second, our study is based on an empirical analysis of domain registry policies, aiming to highlight overlooked security risks in domain lifecycle management and raise community awareness. Using publicly available TLD zone files, we identified candidate vulnerable domain names and host objects. However, this data-driven approach cannot fully capture the diverse policies and operations of all registries and registrars, nor cover the entire DNS space of over 200 million domains. Achieving more comprehensive and accurate detection will require collaboration from the community, especially major domain registries and registrars.

Third, the domain name space currently contains over 200 million domains, and we cannot accurately analyze the status of each domain. However, our analysis provides a lower bound of the threat impact, which is sufficient to underscore the seriousness of the issues. Moreover, after our disclosure to ZDNS, they identified and remediated a large number of relic domains internally, demonstrating that our research contributes to enhancing the security of the DNS ecosystem.

## 9 Conclusion

In this work, we have conducted an in-depth exploration of domain registry operation practices from the perspective of public data, covering 7.9 billion historical WHOIS records and zone files of 1,109 TLDs. Our systematic analysis uncovers three novel deficiencies—twin domain creation, siloed

domain delegation, and relic domain deletion—which introduce security risks across the domain lifecycle. We have confirmed that adversaries are misusing the protective measures of registries to propagate fraudulent websites using twin domains. Furthermore, we discovered that registries' siloed DNS host operation practices have led to a multitude of stale glue records, providing adversaries with a rich array of attack vectors. Moreover, we disclosed a flawed registry operation in the domain deletion process that can prevent domain names from being removed from the domain name space, affecting 50% of TLDs, including `.app`, `.dev`, and `.top`, and leading registry backends, like Identity Digital and GoDaddy. Our work has demonstrated that operation deficiencies in registries are prevalent and urgently require the community's attention.

## Acknowledgments

We sincerely thank all anonymous reviewers and our shepherd for their valuable comments on improving the paper. We also thank the staff at ZDNS and Nomulus for helping us identify and resolve this issue. This work is in part supported by the National Natural Science Foundation of China (62102218) and Zhongguancun Laboratory.

## Ethics Considerations

Our analysis involves using public datasets and querying domain nameservers, with careful consideration of ethical issues in the experimental design. We adhere to ethical standards based on the Menlo Report [35], best practices for network measurement [46], and recommendations for using public dataset [17]. *First*, we collected data from publicly accessible TLD zone files, which include domain delegation information commonly used in DNS research [16, 33, 54], without registrant information. All data was stored securely on internal servers with no unauthorized access. *Second*, for historical WHOIS data, we focused on registration dates and status fields, excluding registrant information to ensure anonymity and privacy. *Third*, we adhered to vendor-imposed rate limits when accessing registrar interfaces to determine domain status and avoid straining their systems. *Forth*, during active DNS probes, such as querying NS records, we strictly limit the scan rate to minimize the impact on public DNS resolvers and nameservers.

## Open Science

*First*, the TLD zone files and registrar interfaces we used are publicly accessible. We will provide the official website and API descriptions. *Second*, we leveraged the historical WHOIS data maintained by our collaborating security company, which cannot be released. However, regular users can still collect the current registration data of domain names through public WHOIS interfaces [2] or use other commercial WHOIS datasets. We will open-source our WHOIS collection script [3]. *Third*, we cannot directly provide a complete list of affected domain name entities due to the standard requirements for vulnerability disclosure; instead, we can offer specific examples that were vulnerable for validation.

## References

[1] gTLD Registry Agreements. https://www.icann.org/en/registry-agreements.

[2] ICANN Lookup. https://lookup.icann.org/en.

[3] Identity Digital. https://www.identity.digital/.

[4] Registration Data Access Protocol (RDAP). https://www.icann.org/rdap.

[5] Repository of IDN practices. https://www.iana.org/domains/idn-tables.

[6] Teleinfo. https://www.teleinfo.cn/.

[7] The Centralized Zone Data Service (CZDS). https://czds.icann.org/home.

[8] ZDNS. https://www.zdns.cn/.

[9] Report on Chinese Variants in Internationalized Top-Level Domains. https://archive.icann.org/en/topics/new-gtlds/chinese-vip-issues-report-03oct11-en.pdf, 2011.

[10] Life cycle of a typical gTLD domain name. https://www.icann.org/resources/pages/gtld-lifecycle-2012-02-25-en/, 2012.

[11] EPP status codes | what do they mean, and why should I know? https://www.icann.org/resources/pages/epp-status-codes-2014-06-16-en, 2014.

[12] Recommendations for Managing Internationalized Domain Name Variant Top-Level Domains Published. https://www.icann.org/en/announcements/details/recommendations-for-managing-internationalized-domain-name-variant-top-level-domains-published-5-2-2019-en, 2019.

[13] IDN Variant Labels. https://icannwiki.org/IDN_Variant_Labels, 2022.

[14] SAC120: SSAC Input to GNSO IDN EPDP on Internationalized Domain Name Variants. https://itp.cdn.icann.org/en/files/security-and-stability-advisory-committee-ssac-reports/sac-120-en.pdf, 2022.

[3] https://zenodo.org/records/14760367

[15] Gautam Akiwate, Mattijs Jonker, Raffaele Sommese, Ian D. Foster, Geoffrey M. Voelker, Stefan Savage, and kc claffy. Unresolved issues: Prevalence, persistence, and perils of lame delegations. In *IMC*, 2020.

[16] Gautam Akiwate, Stefan Savage, Geoffrey M. Voelker, and Kimberly C. Claffy. Risky BIZness: risks derived from registrar name management. In *IMC*, 2021.

[17] Mark Allman and Vern Paxson. Issues and etiquette concerning use of shared measurement data. In Constantine Dovrolis and Matthew Roughan, editors, *IMC*, 2007.

[18] Eihal Alowaisheq, Siyuan Tang, Zhihao Wang, Fatemah Alharbi, Xiaojing Liao, and XiaoFeng Wang. Zombie Awakening: Stealthy hijacking of active domains through DNS hosting referral. In *CCS*, 2020.

[19] Mark Andrews, Shumon Huque, Paul Wouters, and Duane Wessels. DNS glue requirements in referral responses. *RFC*, 9471, 2023.

[20] Bohdan Turkynevych. Domains Project: Processing petabytes of data so you don't have to. https://domainsproject.org/.

[21] Leslie Daigle. WHOIS protocol specification. *RFC*, 3912:1–4, 2004.

[22] Dynadot. What are TrueName domains? https://www.dynadot.com/community/help/question/truename-domains.

[23] Eclypsium. Ducks Now Sitting (DNS): Internet Infrastructure Insecurity. https://eclypsium.com/blog/ducks-now-sitting-dns-internet-infrastructure-insecurity/, 2024.

[24] GoDaddy. Domain Name Registry Services - GoDaddy Registry. https://registry.godaddy/.

[25] Google Registry. Expanding the Online Frontier | Google Registry. https://www.registry.google/.

[26] greenSec. new gTLD Statistics - Get detailed insights about TLDs. https://www.ntldstats.com/.

[27] Scott Hollenbeck. Extensible provisioning protocol (EPP). *RFC*, 5730:1–67, 2009.

[28] Scott Hollenbeck. Extensible provisioning protocol (EPP) domain name mapping. *RFC*, 5731:1–44, 2009.

[29] Scott Hollenbeck. Extensible provisioning protocol (EPP) host mapping. *RFC*, 5732:1–29, 2009.

[30] Hang Hu, Steve T. K. Jan, Yang Wang, and Gang Wang. Assessing browser-level defense against idn-based phishing. In *USENIX Security 2021*, pages 3739–3756, 2021.

[31] ICANN-IDN ccTLD Fast Track Program. Proposed implementation details regarding: Development and use of idn tables and character variants for second and top level strings. https://www.icann.org/en/system/files/files/proposed-implementation-details-idn-tables-revision-1-clean-29may09-en.pdf.

[32] Indiana JSON. Can I Take Over DNS? A list of DNS providers and whether their zones are vulnerable to DNS takeover. https://github.com/indianajson/can-i-take-over-dns, 2023.

[33] Andrew J. Kalafut, Minaxi Gupta, Christopher A. Cole, Lei Chen, and Nathan E. Myers. An empirical study of orphan dns servers in the internet. In *IMC*, 2010.

[34] Aqsa Kashaf, Vyas Sekar, and Yuvraj Agarwal. Analyzing Third Party Service Dependencies in Modern Web Services: Have We Learned from the Mirai-Dyn Incident? In *Proceedings of the ACM Internet Measurement Conference*. Association for Computing Machinery, 2020.

[35] Erin Kenneally and David Dittrich. The Menlo report: Ethical principles guiding information and communication technology research. *Available at SSRN 2445102*, 2012.

[36] Tobias Lauinger, Abdelberi Chaabane, Ahmet Salih Buyukkayhan, Kaan Onarlioglu, and William Robertson. Game of registrars: An empirical analysis of Post-Expiration domain name takeovers. In *26th USENIX Security Symposium (USENIX Security 17)*, August 2017.

[37] Xiaodong Lee, Wei Mao, Erin Chen, Nai-Wen Hsu, and John C. Klensin. Registration and administration recommendations for chinese domain names. *RFC*, 4713:1–9, 2006.

[38] John R. Levine and Paul E. Hoffman. Variants in second-level names registered in top-level domains. *RFC*, 6927:1–18, 2013.

[39] Baojun Liu, Chaoyi Lu, Zhou Li, Ying Liu, Hai-Xin Duan, Shuang Hao, and Zaifeng Zhang. A reexamination of internationalized domain names: The good, the bad and the ugly. In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2018.

[40] Daiping Liu, Shuai Hao, and Haining Wang. All your dns records point to us: Understanding the security threats of dangling dns records. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, 2016.

[41] miHoYo. Genshin impact – step into a vast magical world of adventure. https://genshin.hoyoverse.com/.

[42] Paul V. Mockapetris. Domain names - concepts and facilities. *RFC 1034*, 1987.

[43] Giovane C. M. Moura, Sebastian Castro, Wes Hardaker, Maarten Wullink, and Cristian Hesselman. Clouding up the internet: how centralized is DNS traffic becoming? In *IMC*, 2020.

[44] Neal S. The Watchtowr DNS WHOIS Vulnerability: A Warning for the RDAP Transition and Future Risks. https://www.linkedin.com/pulse/watchtowr-dns-whois-vulnerability-warning-rdap-transition-neal-smyth-xklve.

[45] Nominet. The official registry for UK domain names. https://www.nominet.uk/.

[46] Craig Partridge and Mark Allman. Ethical considerations in network measurement papers. *Commun. ACM*, 59(10):58–64, sep 2016.

[47] Sally Costerton. ICANN and Contracted Parties Negotiate About Improved DNS Abuse Requirements. https://www.icann.org/en/blogs/details/icann-and-contracted-parties-negotiate-about-improved-dns-abuse-requirements-18-01-2023-en, 2023.

[48] Raffaele Sommese, Mattijs Jonker, Roland van Rijswijk-Deij, Alberto Dainotti, Kimberly C. Claffy, and Anna Sperotto. The forgotten side of DNS: orphan and abandoned records. In *EuroS&P Workshops*, 2020.

[49] Hiroaki Suzuki, Daiki Chiba, Yoshiro Yoneya, Tatsuya Mori, and Shigeki Goto. Shamfinder: An automated framework for detecting IDN homographs. In *Proceedings of IMC*, pages 449–462, 2019.

[50] The hackcompute group. Can I speak to your manager? hacking root EPP servers to take control of zones. https://hackcompute.com/hacking-epp-servers/, 2023.

[51] Fenglu Zhang, Yunyi Zhang, Baojun Liu, Eihal Alowaisheq, Lingyun Ying, Xiang Li, Zaifeng Zhang, Ying Liu, Haixin Duan, and Min Zhang. Wolf in sheep's clothing: Evaluating security risks of the undelegated record on DNS hosting services. In *IMC*, 2023.

[52] Mingming Zhang, Xiang Li, Baojun Liu, Jianyu Lu, Yiming Zhang, Jianjun Chen, Haixin Duan, Shuang Hao, and Xiaofeng Zheng. Detecting and measuring security risks of hosting-based dangling domains. *Proc. ACM Meas. Anal. Comput. Syst.*, 7(1), 2023.

[53] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, Yan Shoshitaishvili, Adam Doupé, and Gail-Joon Ahn. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In *42nd IEEE Symposium on Security and Privacy, 2021*, pages 1109–1124, 2021.

[54] Yunyi Zhang, Baojun Liu, Haixin Duan, Min Zhang, Xiang Li, Fan Shi, Chengxi Xu, and Eihal Alowaisheq. Rethinking the Security Threats of Stale DNS Glue Records. In *Proceedings of the 32nd USENIX Security Symposium*, 2024.

[55] Yunyi Zhang, Chengxi Xu, Fan Shi, Miao Hu, Min Zhang, Yuwei Li, and Zhijie Xie. Understanding and characterizing the adoption of internationalized domain names in practice. *IEEE Transactions on Dependable and Secure Computing*, 2024.

[56] Yunyi Zhang, Mingming Zhang, Baojun Liu, Zhang Liu, Jia Zhang, Haixin Duan, Min Zhang, Fan Shi, and Chengxi Xu. Cross the Zone: Toward a Covert Domain Hijacking via Shared DNS Infrastructure. In *Proceedings of the 32nd USENIX Security Symposium*, 2024.

## A Common Domain Registry Operations and Domain Lifecycle

ICANN defines various stages of a domain's lifecycle [10], each associated with specific EPP status codes. Figure 7 demonstrates the lifecycle stages and common EPP status transitions triggered by registry operations, according to ICANN's documentations [10, 11]. During the domain creation period, a registry conducts checks on registrant information and domain status before creating the domain object and adding its delegation records to the TLD zone file, as shown in Figure 12. This process transitions the domain's lifecycle from `Available` to `Registered`. During the domain management period, the registry is responsible for monitoring domain and delegation statuses and updating TLD zone files based on registrar commands or automated programs. It manages the addition and update of NS records for domains under its jurisdiction. During the domain deletion period, the registry must verify domain dependencies (e.g., whether other domains rely on it) and statuses (e.g., whether the domain is taken down or prohibited from deletion), then securely remove all associated domain objects and delegation records from the TLD zone.

## B Case Study of Abused Domain Names

**Example 1:** `xn--4gq220j14gckc.top` (一谕终见.top) and `xn--4gqz56iuyholb.top` (一諭終見.top). Through WHOIS, we determined that the domain name

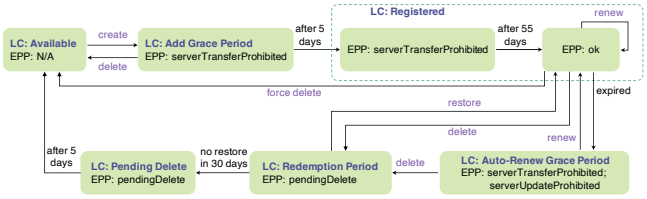


Figure 7: Registry operations and EPP status transitions across the domain lifecycle. Operations like `create`, `delete`, and `renew` are carried out by registries, triggered either by registrars or automated programs. The figure illustrates only the common standard lifecycle stages.


Figure 8: Webpage of `xn--4gq220j14gckc.top`.


Figure 9: Webpage 1 of `xn--4gqz56iuyholb.top`.

`xn--4gq220j14gckc.top` was registered by the registrant on Alibaba Cloud in 2023. However, we could not find registration information for the domain name `xn--4gqz56iuyholb.top`, yet it has a delegation record in the zone file, hence, we consider it a *twin domain*. Figure 8 displays the webpage of the domain name `xn--4gq220j14gckc.top`, which is a gaming page. However, the webpage of the domain `xn--4gqz56iuyholb.top` is completely different and contains evident fraudulent information. Figure 9 shows the page as it appeared during our first visit, cluttered with numerous data and embedded links to other underground websites. Figure 10, on the other hand, displays the page after a month, which has been replaced with a lottery scam page.

**Example 2:** `xn--8mrwhx77h.top` (原神号.top) and `xn--8mr619fm3i.top` (原神號.top). The domain name


Figure 10: Webpage 2 of `xn--4gqz56iuyholb.top`.


Figure 11: Webpage of `xn--8mrwhx77h.top`.

`xn--8mrwhx77h.top` is a website for sharing Genshin Impact accounts, as shown in Figure 11. Genshin Impact is a globally popular open-world ARPG (Action Role-Playing Game) [41]. However, the domain name `xn--8mr619fm3i.top` appears to be unregistered and points to the same lottery scam page as Figure 10.

## C An Example of Glue Record Injection

To inject stale glue records into the target resolver, the attacker first registered the domain name `attacker.example` and then configured its NS records to correspond to the stale glue records (i.e., `ns1.foo.example`), as shown in Figure 12. The attacker then initiates a resolution request for `attacker.example` to the target resolver. When the target resolver sends a query to the authoritative server of `.example`, the authoritative server includes the stale glue records in the referral response. Ultimately, the target resolver caches and utilizes the stale glue records.

```
;; .example zone file
attacker.example      NS   ns1.foo.example
;;stale glue record
ns1.foo.example   A    cloud ip
```

Figure 12: An example of resource record configuration to exploit stale glue records.