Cross the Zone: Toward a Covert Domain Hijacking via Shared DNS Infrastructure

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

https://www.usenix.org/conference/usenixsecurity24/presentation/zhang-yunyi-zone

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Cross the Zone: Toward a Covert Domain Hijacking via Shared DNS Infrastructure

Yunyi Zhang [†‡], Mingming Zhang [§], Baojun Liu[‡§✉], Zhan Liu[‡], Jia Zhang[‡], Haixin Duan[‡§]
Min Zhang[†✉], Fan Shi[†], Chengxi Xu[†]

[†]National University of Defense Technology, [‡]Tsinghua University, [§]Zhongguancun Laboratory

Abstract

Domain Name System (DNS) establishes clear responsibility boundaries among nameservers for managing DNS records via authoritative delegation. However, the rise of third-party public services has blurred this boundary. In this paper, we uncover a novel attack surface, named *XDAuth*, arising from public authoritative nameserver infrastructure's failure to isolate data across zones adequately. This flaw enables adversaries to inject arbitrary resource records across logical authority boundaries and covertly hijack domain names without authority. Unlike prior research on stale NS records, which concentrated on domain names delegated to expired nameservers or those of hosting service providers, XDAuth targets enterprises that maintain their authoritative domain names. We demonstrate that XDAuth is entirely feasible, and through comprehensive measurements, we identify 12 potential vulnerable providers (e.g., Amazon Route 53, NSONE, and DigiCert DNS), affecting 125,124 domains of notable enterprises, including the World Bank, and the BBC. Moreover, we responsibly disclose the issue to the affected vendors. Some DNS providers and enterprises (e.g., Amazon Route 53) have recognized the issue and are adopting mitigation measures based on our suggestions.

1 Introduction

Domain Name System (DNS), as a fundamental Internet infrastructure, is pivotal for translating domain names to IP addresses, supporting mainstream Internet services like email and CDNs [15, 16, 33]. It organizes the global namespace into distinct zones using a hierarchical structure, with authoritative nameservers maintaining each zone. Once the authoritative nameservers are compromised, all domain names delegated to them will be hijacked, causing serious cybercrimes (e.g., certificate forging, APT attacks) [4, 35, 38, 53]. More and more enterprises are turning to professional DNS hosting services for more robust and secure authoritative services.

✉ Corresponding author.

In recent years, researchers have uncovered security implications of stale domain delegations (StaleNS), identifying cases where domains are pointed to vulnerable DNS hosting providers lacking domain ownership validation (DOV) [6, 42] or their NS domains have expired [2, 35]. To prevent attackers from *claiming or manipulating StaleNS domains*, a series of responsible providers like Amazon Route 53 and Cloudflare have implemented various defensive strategies, such as tracking NS allocation [8] and performing TXT-based DOV [23]. However, given the complexity of cloud hosting scenarios and the diversity of DNS architecture, we wonder about the effectiveness of these measures in ensuring comprehensive protection. *Is it safe if the NS domains have not expired and do not belong to a vulnerable provider?*

The DNS protocol mandates the isolation of data between different zones within authoritative nameservers, ensuring a nameserver cannot exceed its designated authority over domain names. Meanwhile, enterprises expect their DNS zone data on third-party hosting platforms to be inaccessible and can not be tampered with by unauthorized parties. Thus, some enterprises use self-controlled NS domains to establish nameservers via public hosting providers' DNS infrastructures, preventing adversaries from reusing dangling NS. For example, Nike's NS domain is `ns-v#.nike.com`, but its infrastructure relies on Amazon Route 53 DNS infrastructure. However, we observed that *service providers' DNS infrastructures often fail to properly enforce DNS zone isolation for customers*. As a result, both the enterprises' and the providers' nameservers share the same DNS databases, which are called *shared nameservers*. This results in the enterprise's DNS zone not being fully under its control, opening the door for attackers to modify DNS resource records and potentially hijack domains.

Our study. In this paper, we propose a novel threat model named Cross Domain Authority Boundary (XDAuth). An attacker could inject arbitrary resource records covertly for a victim's domain name by exploiting an *out-of-delegation* nameserver. For instance, a customer deploys their authoritative nameserver (e.g., *ns.c1.com*) leveraging a provider's DNS infrastructure. Due to the absence of effective delegation records, unauthorized claims on DNS hosting service providers can only be abused through specific utilization scenarios [59]. Thus, most DNS hosting providers still lack effective domain name ownership verification strategies, allowing legitimate users to claim well-known domain names freely. Moreover, since the lack of DNS zone isolation in the provider, the attacker can manipulate the resource records of domain names delegated to *ns.c1.com* through the nameserver (e.g., *ns.provider.com*) of the provider. Unlike StaleNS, the threat is hard to detect since the exploited nameserver is not expired and is outside the delegation relationship.

We have conducted an empirical study and proved that XDAuth is completely realistic. We successfully implemented cross-authoritative nameserver domain takeover at various leading DNS hosting providers. To evaluate the prevalence of XDAuth, we proposed a semi-automated detection framework, named XDAuthChecker, to uncover XDAuth threats in the wild effectively. We used the framework to explore nameserver dependencies and identify shared nameserver groups systematically. For each group, we examined the existence of vulnerable hosting providers that can be exploited to inject forged DNS records into the shared nameservers. Subsequently, we conducted a large-scale measurement study of the DNS-sharing ecosystem and the enterprises that XDAuth affects.

Findings. We revealed that shared nameservers are indeed *widespread and severe* by running XDAuthChecker on 1,090 gTLD zone files. We identified a total of 2,372 shared nameserver groups, consisting of 60,974 nameservers with identical IP addresses (SharedAuth-I) and 4,800 nameservers with varied NS domains and IP addresses (SharedAuth-II). Upon analyzing these shared groups, we identified 12 potential vulnerable providers, including *Amazon Route 53*, *NSONE*, and *Digicert DNS*. These providers indirectly affect 1,881 other nameservers, with 981 of them ranking in the Tranco top 1M [45], e.g., *ns-v1.nike.com*, highlighting a substantial security threat. After detecting domains delegated to affected nameservers, we found that XDAuth poses security risks to numerous well-known enterprises. As a result, we have discovered 125,124 domains vulnerable to XDAuth attacks, encompassing notable entities like *McKesson*, and *Canon*. The affected entities also include domain management or digital certificate companies, indicating their customer domains are susceptible to domain hijacking. For instance, we confirmed that 184 domains managed by *Sectigo* are affected.

Mitigation and responsible disclosure. We offer feasible suggestions for vulnerable providers, like tracking global domain hosting status in shared nameservers. We have also reported the issues to the affected hosting providers and enterprises. So far, NSONE, *Anon.C* [1], and Amazon Route 53 have acknowledged the issue, with *Anon.C* currently implementing our proposed mitigation. Moreover, *Anon.E* [2] and McKesson have also acknowledged the vulnerability and fixed the affected domains.

[1]According to the company's requirements, we do not disclose the company's name here. Throughout this paper, it will be referred to by the *Anon.C*.

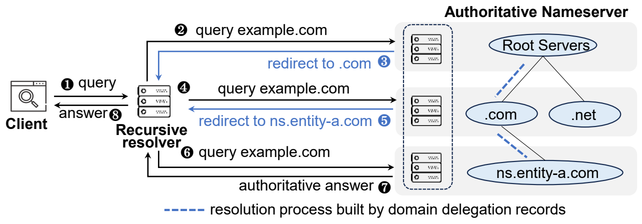


Figure 1: Domain resolution process.

Contributions. We make the following contributions:

- *New attack surface.* We uncover a new attack surface in the DNS infrastructure: shared nameservers infrastructure. To our knowledge, this is the first public disclosure of such a vulnerability, and our extensive measurement reveals that shared authoritative nameservers are highly prevalent.
- *Novel methodology and findings.* We propose XDAuthChecker, a novel approach to discovering shared nameserver threats. Our results demonstrate that XDAuth can circumvent current best protective measures, enabling covert out-delegation domain hijacking and affecting some well-known enterprises.
- *Responsible disclosure.* We responsibly disclose the threat to affected providers and enterprises, provide practical mitigation, and have received their confirmation.

2 Background and Related Work

2.1 Domain Resolution Process

The Domain Name System (DNS) is a hierarchical structure as shown in Figure 1. The authoritative DNS servers (also named nameservers or NS) of parent domains maintain the delegation records for their subordinate domains. For example, root servers hold `NS` records for all Top-Level Domains (TLDs), and nameservers for `.com` maintain `NS` records for all Second-Level Domains (SLDs, commonly referred to as apex domains) under `.com`, such as the NS domain `ns.entity-a.com` for `example.com`. To obtain a domain's IP address, a client initiates a domain query to recursive resolvers (step ❶). The recursive resolvers fetch the domain delegation information (`NS` and `glue` records[3]) from nameservers at different levels, constructing the domain resolution path (step ❷ ∼ ❼). Finally, an authoritative response is obtained from the nameservers of the queried domain (step ❼).

The authoritative nameserver is designed to offer authorized resolution services only for the domains delegated to it

[2]A Fortune 500 company has asked us not to disclose any of their information. Throughout this paper, it will be referred to by the *Anon.E*.

[3]A `glue` record provides the IP address of a nameserver at a domain name registry.

(domain delegation) [39, 58]. Therefore, even though there may be multiple IPs behind a nameserver, the resolution path of a domain name is visible from a namespace perspective. The fact we can determine is that `ns.entity-a.com` can provide authoritative responses for `example.com`, while other nameservers cannot. However, with the prevalence of DNS hosting services, the boundary between authoritative nameservers is no longer clear. When different authoritative nameservers are combined into a large service platform, the resolution path of a domain name may become unpredictable.

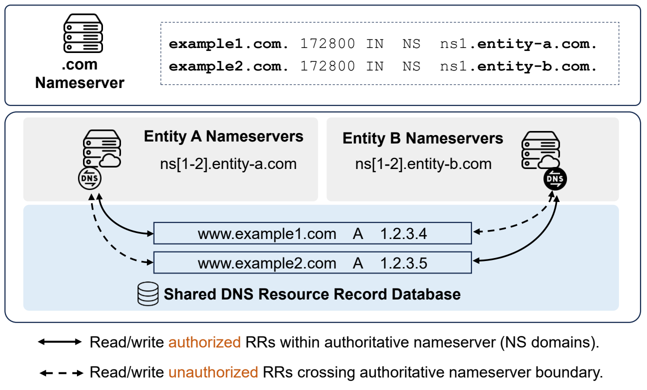
2.2 Nameserver-based Domain Hijacking

Authoritative nameservers maintain all resource records (RRs) of authorized domain names. By controlling a nameserver, an attacker can manipulate or inject RRs for a domain name. Previous work has shown some domain hijacking threats by controlling the victim domain's nameservers.

DNS infrastructure tampering. A common scenario involves an adversary compromising a victim domain holder's account at their registrar or the registrar platform, gaining the ability to manipulate the domain's DNS settings [11]. Such an attack became widely known through reports by Cisco Talos [56] and FireEye's Mandiant [41]. This led the US DHS to issue an emergency directive for implementing mitigation measures in government systems. Researchers have leveraged the behavioral patterns and historical data of domain names to identify authoritative servers that are no longer under proper control [4, 36]. While this type of attack poses a significant threat, compromising a well-managed authoritative nameserver remains challenging.

StaleNS threats. StaleNS occurs when a domain's nameserver stops providing authoritative services for it. Prior work mainly focused on scenarios where domain names are delegated to either *expired NS domains* or nameservers of hosting services that have been discontinued (*discontinued hosting*). Liu et al. [35] showed that attackers could take over domain names by registering expired NS domains if domain owners forget to remove stale NS records. This vulnerability extends to TLDs as well [2, 12, 20, 38, 61]. Additionally, Akiwate et al. [3] illustrated that expired NS domains could also result from the improper operations of domain registrars.

Domain name configuration has become increasingly complex with the advent of DNS hosting services. Registrants are at increased risk of misconfigurations, like failing to update authoritative nameservers after canceling domain hosting. In this scenario, attackers could hijack dehosted customer domains with unchanged delegations by rehosting them [30, 37, 50, 60]. Even if registrants have removed stale NS records at the TLD level, NS records may still remain on the nameservers of SLDs, which could result in ZAW attacks [6]. In addition to hijacking customer domains, the implementation flaws in DNS hosting providers also enable an attacker to take over the providers' domains [49].



.com Nameserver
example1.com. 172800 IN NS ns1.entity-a.com.
example2.com. 172800 IN NS ns1.entity-b.com.
Entity A Nameservers
ns[1-2].entity-a.com
Entity B Nameservers
ns[1-2].entity-b.com
DNS
www.example1.com A 1.2.3.4
www.example2.com A 1.2.3.5
Shared DNS Resource Record Database
Read/write authorized RRs within authoritative nameserver (NS domains).
Read/write unauthorized RRs crossing authoritative nameserver boundary.

Figure 2: SharedAuth: **Shared Auth**oritative nameserver.

In response to such issues, service providers like Amazon Route 53 have initiated mitigation strategies, such as randomizing NS allocations and preventing the reuse of NS for the same domain, thwarting potential domain takeovers [8].

Comparison with previous threats. Similar to StaleNS, XDAuth aims to hijack prominent domain names and manipulate their resolution results. However, XDAuth involves scenarios where attackers can not take control of the victim domain's nameservers, which *remain unexpired and under the active management of the victim enterprise itself.* XDAuth hijacks the victim domain by injecting crafted DNS records for the domain into the shared nameserver infrastructure, instead of directly controlling the domain's nameservers.

Prior research on StaleNS, notably [6] and [60], focuses on domain names delegated to hosting providers' NS domains, overlooking scenarios where enterprises maintain their own NSs. XDAuth fills this gap by showing that domains with NS that can not be allocated via hosting providers and remains active also face the risk of takeover.

3 Overview of XDAuth Attack

3.1 Shared Authoritative Nameservers

Definition of SharedAuth. We define shared authoritative nameservers (abbreviated as SharedAuth) as different nameservers that share a DNS resource record (RR) database or a DNS zone either due to a common underlying DNS infrastructure or through zone transfer. These nameservers may be operated by cloud hosting providers or self-controlled by organizations or companies. In Figure 2, we showcase a schema for SharedAuth. Entity A and B utilize `ns[1-2].entity-a.com` and `ns[1-2].entity-b.com`, respectively, to provide authoritative DNS records for their managed domain names such as `example1.com` and `example2.com`. Despite differences in their NS domains, we consider the nameservers of Entity A and B as SharedAuth, as all DNS RRs for both `example1.com` and `example2.com` are stored in the same DNS RR database.

In this paper, **we highlight the lack of DNS data isolation among entities sharing the same DNS infrastructure in practice**. As per RFC 1034 [39], domain nameservers are authorized to hold zones and provide authoritative answers only for their delegated domain names. For instance, according to the DNS setting in the `.com` zonefile (the top box in Figure 2), `ns[1-2].entity-a.com` should handle queries only for `example1.com`, not `example2.com`. However, the adoption of SharedAuth by both entities compromises these authority boundaries, and there is no restriction for any entity to access resources from the shared DNS database by design. As a result, their nameservers can manage domain names beyond their authorized scope. For example, someone can configure or fetch DNS records for `example2.com` through `ns[1-2].entity-a.com`, as shown by the dashed lines in Figure 2. Due to the lack of proper access control in the shared DNS database, the stored DNS records are vulnerable to poisoning or tampering.

Scenarios for deploying SharedAuth. SharedAuth is often deployed in cloud hosting environments by two or more entities with the following scenarios:

- *Hosting and non-hosting*. Some organizations, like governments and companies, may seek dependable DNS infrastructures from public hosting providers. In this scenario, we observe that many hosting providers offer customization services like vanity DNS [13, 14, 18, 47] and white-label nameservers [9]. They initialize hosting instances from a shared DNS infrastructure, enabling customers to manage their own DNS zones with their self-controlled NS domains. This practice has been adopted by several renowned companies, including *Anon.E*, Canon, and Nike.

- *Hosting and hosting*. Sharing nameservers can also result from business collaborations and service upgrades. First, small hosting providers or intermediaries opt to employ established nameservers of prominent hosting service suppliers instead of constructing brand-new DNS infrastructure. For example, some web hosting providers (e.g., Squarespace) share nameservers with a well-known DNS provider, NSONE. Second, hosting providers may update their DNS infrastructure and change NS domains as part of service enhancements. To ensure seamless migration, old NS domains may continue to be used, with DNS resource records synchronized between the old and new nameservers.

These practices indicate that SharedAuth has been widely adopted. However, ensuring data isolation between different DNS zones is crucial to avoid potential security risks, such as data reuse or alteration, when nameservers are shared. In Section 5.2, we will further explore the prevalence and scenarios of nameserver sharing in the real world.

Types of SharedAuth. To clarify further, let's consider a SharedAuth pool containing two groups of NS domains, NS1 and NS2. Based on their apex domains and IP addresses, all SharedAuth can be classified into four types (see Figure 3):

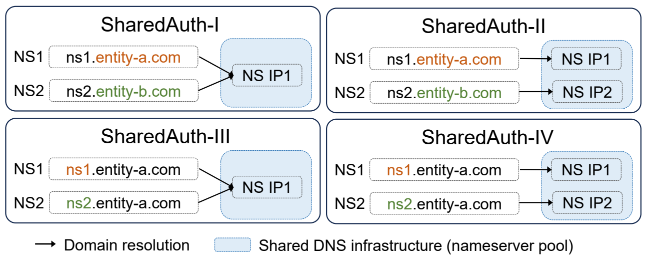


Figure 3: Classification of SharedAuth.

- *SharedAuth-I*. NS1 and NS2 belong to different apex domains but share the same IP address.
- *SharedAuth-II*. NS1 and NS2 belong to different apex domains and have distinct IP addresses. Meanwhile, they facilitate cross-resolution, namely, a domain name hosted on NS2 can be resolved through NS1.
- *SharedAuth-III*. NS1 and NS2 are different subdomains under the same apex domain and share the same IP address.
- *SharedAuth-IV*. NS1 and NS2 are different subdomains under the same apex domain and have different IPs.

Note that our discussion of SharedAuth excludes scenarios where NS domains are configured with canonical names (CNAMEs), because DNS standards prohibit such associations [21]. Furthermore, the identification for these four types of SharedAuth differs. For instance, NS domains in SharedAuth-III and SharedAuth-IV share the same apex domain, making them relatively easy to collect. However, there appears to be no correlation among the NS domains in SharedAuth-I and SharedAuth-II, rendering them more covert and challenging to discover. Moreover, they would result in the exposure of two entities that are unaware of each other to significant security risks. Consequently, we will focus our effort more on SharedAuth-I and SharedAuth-II in Section 4.

3.2 Threat model

Based on the SharedAuth scenario, we introduce a novel domain takeover threat model termed Cross Domain Authority Boundary (XDAuth). It targets prominent domain names delegated to secure nameservers with proper domain ownership validation (DOV). For a clear description, we outline five participants in the XDAuth model (depicted in Figure 4):

- A secure-entity that can be either a hosting provider or a non-hosting organization (e.g., companies, authorities, and government agencies). This entity is responsible for managing all authorized DNS RRs for delegated domain names through carefully maintained nameservers, identified as `ns[1-2].secure.com`. These nameservers either implement stringent DOV strategies or do not offer public hosting services, making them highly resilient to StaleNS attacks [6].

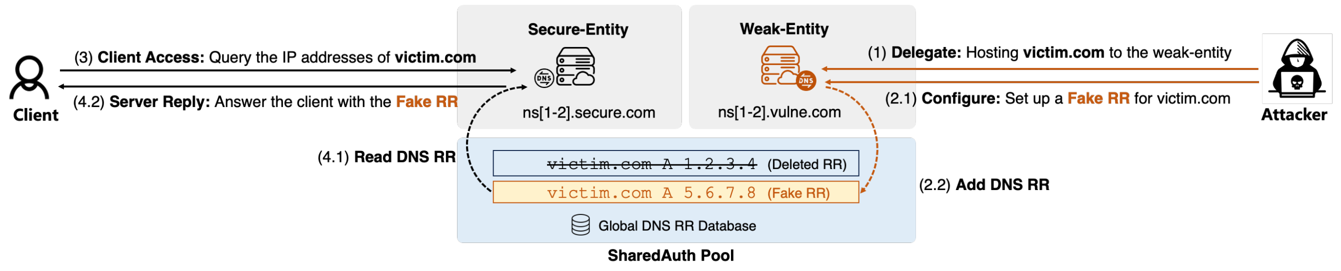

Secure-Entity
Weak-Entity
Client
Attacker
(3) Client Access: Query the IP addresses of victim.com
(4.2) Server Reply: Answer the client with the Fake RR
ns[1-2].secure.com
ns[1-2].vulne.com
(1) Delegate: Hosting victim.com to the weak-entity
(2.1) Configure: Set up a Fake RR for victim.com
(4.1) Read DNS RR
~~victim.com A 1.2.3.4~~ (Deleted RR)
victim.com A 5.6.7.8 (Fake RR)
(2.2) Add DNS RR
Global DNS RR Database
SharedAuth Pool

Figure 4: Threat model of Cross Authority-Boundary Attack (XDAuth). Both secure-entity and weak-entity utilize the global DNS database to manage RRs. `victim.com`'s domain owner can configure or delete its DNS RRs, deleting one (with strikethrough) from the database. Meanwhile, the attacker can configure arbitrary RRs in the DNS database to hijack `victim.com`.

- A weak-entity could be a public hosting provider that has security flaws in DOV, which allows the unauthorized claim of arbitrary domain names. It supports authoritative domain resolution services through `ns[1-2].vulne.com` and shares the underlying DNS infrastructure, such as the RR database, with secure-entity.

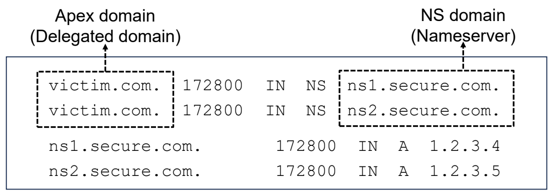
- A victim domain name, `victim.com`, that is delegated to the nameservers controlled by secure-entity, whose nameserver setting is shown in Figure 5. However, these nameservers have stopped the service to it, responding `REFUSED`, which is a common occurrence in the DNS ecosystem [2, 51, 60].

- An attacker can be a legitimate customer of weak-entity and can deploy any domain names on weak-entity without compromising the DNS system or any servers. The attacker aims to covertly hijack `victim.com` by exploiting the SharedAuth pool between weak-entity and secure-entity.

- A client that attempts to access `victim.com`.

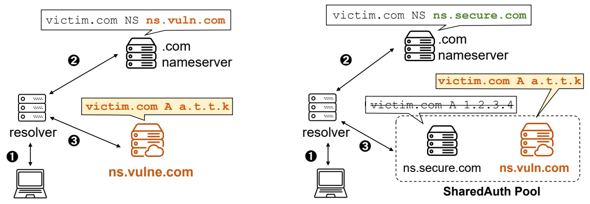
To launch an XDAuth attack, attackers are required to follow two requirements. First, they need to uncover the SharedAuth pool that includes the victim domain's nameservers, namely, identifying the nameservers shared with the victim domain. We introduce a black-box testing method to explore nameserver dependencies behind the DNS infrastructure in Section 4. Our results demonstrate that SharedAuth has been utilized by a number of large enterprises on a significant scale. Second, they should gain access to a vulnerable nameserver (e.g., `ns[1-2].vulne.com`) in that SharedAuth pool to configure arbitrary DNS RRs for the victim domain. This condition can be easily achieved by leveraging public DNS hosting services, since previous research [6, 59] demonstrated that most DNS hosting providers lack proper DOV policies, allowing users to claim arbitrary domain names from the providers.

Attack flow. Figure 4 provides a schematic overview of an XDAuth attack. Suppose the domain owner of `victim.com`



```
victim.com.  172800  IN  NS  ns1.secure.com.
victim.com.  172800  IN  NS  ns2.secure.com.
ns1.secure.com.     172800  IN  A  1.2.3.4
ns2.secure.com.     172800  IN  A  1.2.3.5
```

Figure 5: Example of .com TLD zone file information.



(a) StaleNS: The `victim.com` was delegated to `ns.vulne.com`, which **has expired or belongs to a vulnerable hosting provider**. After acquiring `ns.vulne.com`, attackers can take over `victim.com` and configure arbitrary resource records for it, leading the client to `a.t.t.k`.

(b) XDAuth: The `victim.com` was delegated to `ns.secure.com`, which **remains active in use**. However, attackers can inject crafted RRs through `ns.vuln.com` in the SharedAuth pool. In the end, the client will unwittingly access `a.t.t.k` after querying `victim.com`.

Figure 6: Comparison between StaleNS and XDAuth.

cancels its delegation to secure-entity but retains NS records pointing to `ns[1-2].secure.com`, as depicted in Figure 5. Despite this, these NS domains remain active, thwarting the attacker's attempts to take over `victim.com` by registering or controlling secure-entity' nameservers, as done in previous attacks [6, 35]. In this situation, the attacker could host `victim.com` on weak-entity (step 1) and set up a fake RR for it using weak-entity's nameservers (step 2.1). Meanwhile, these fake DNS RRs will be stored (e.g., `5.6.7.8`) in the DNS database shared by weak-entity and secure-entity (step 2.2). If a client queries `victim.com` (step 3), the victim domain's nameservers, `ns[1-2].secure.com`, would retrieve the fake DNS RRs from the shared database (step 4.1) and respond to the client (step 4.2). Consequently, the client may unwittingly access malicious IPs controlled by the attackers.

Comparing threat scenarios of StaleNS and XDAuth. StaleNS attacks (Figure 6a) target victim domains delegated to discontinued nameservers, leaving stale NS records in the TLD zone. These nameservers, either expired NS domains or those of vulnerable public hosting providers, no longer offer authoritative domain resolution services for the victim domains. Consequently, the attacker can obtain control by re-registering expired NS domains or re-allocating the same NS domains with the victim domains from the hosting provider, using them to take over the victim domain.

Instead, XDAuth attacks (Figure 6b) target victim domains whose nameservers are self-managed by non-hosting organizations or well-managed by public DNS hosting providers. For instance, the domain `victim.com`, delegating to `ns.secure.com` or its subzone `ns.victim.com`, can also be targeted in an XDAuth attack. Unlike StaleNS, the nameservers of the victim domain remain active and are strict in DOV, preventing attackers from acquiring it. XDAuth attackers could exploit nameservers (e.g., `ns.vuln.com`) that share DNS infrastructure with `ns.secure.com` to craft RRs and hijack `victim.com`. In this model, *neither the client nor the domain owner can detect this covert operation, as there are no stale records at either the TLD or SLD levels.*. Thus, it enables offline covert domain hijacking via a novel, previously unreported attack vector.

Discussion. The Internet community has proposed new DNS technologies, such as DNS over HTTPS (DoH), DNS over TLS (DoT), to enhance DNS security, authenticity, and integrity. DoH and DoT only secure the transaction channel, preventing eavesdropping and manipulation in path. However, they cannot prevent XDAuth attacks, as they operate offline from the authoritative nameserver's side by tampering with DNS settings (e.g., SLD zone files) rather than manipulating DNS traffic in transit. Note that a potential limitation of XDAuth is that it primarily affects domain names whose resolutions are suspended, i.e., their nameservers should respond with `REFUSED`. While this condition may seem strict, it is a common scenario in all domain takeover attacks, even in cases of StaleNS. Meanwhile, our analysis reveals such domain names are prevalent, especially among well-known companies (Section 5.3). The inactive domain resolution status and the apparent independence of nameservers may give users a false sense of security, exposing them to XDAuth.

4 Methodology of XDAuth Detection

We introduce **XDAuthChecker**, a framework for evaluating the real-world security implications of XDAuth threats. It aims to (i) detect shared nameservers and potential victim domain names in the wild, and (ii) uncover affected providers and organizations associated with the victim domains. This section provides an overview of the framework's concept and workflow, along with a detailed methodology.

4.1 Overview of XDAuthChecker

Challenges. Designing XDAuthChecker is non-trivial: (C1) The domain name space encompasses over 200M domain names and 2M nameservers, resulting in vast delegation records. Detecting shared nameservers and potential victim domains among this vast pool is overwhelming. (C2) There is no direct evidence for determining the controlling entity of certain nameservers, making it difficult to identify SharedAuth pools based solely on NS domains and NS IPs.

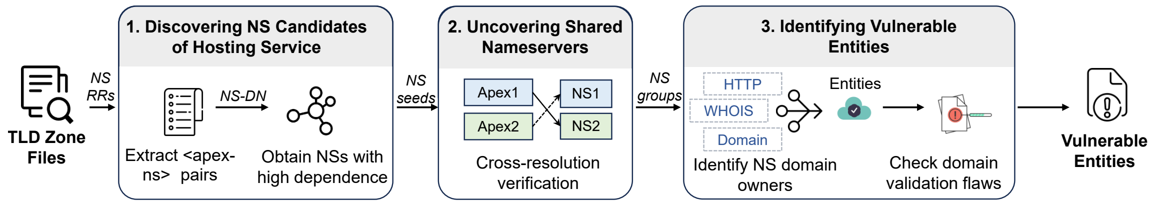
For (C1), we propose NS Dependency Number (*NS-DN*) as the count of apex domains delegating to an NS. We observe NSs with high *NS-DN*s might belong to hosting providers or organizations controlling large-scale domains. Such NSs are more likely to be part of SharedAuth pools and associated with affected organizations than those with lower *NS-DN*s. So we attempt to discover nameserver seeds by limiting *NS-DN*s in Section 4.2. For (C2), we notice that any nameserver within a SharedAuth pool could handle authoritative responses for all delegated domain names. Thus, in Section 4.3, we propose a cross-resolution method to check whether nameservers belong to the same pool.

XDAuthChecker workflow. As depicted in Figure 7, our framework takes TLD zone files as input and contains three primary processes:

In the first module, we discover NS domains potentially linked with public hosting services. Our focus lies on prominent hosting providers, as the nameservers they assign serve numerous customer domains, potentially leading to widespread security implications if vulnerable. We consider the NS domains of these hosting providers as the nameserver seeds for uncovering additional SharedAuth pools, as they represent potential weak points (i.e., weak-entity) for XDAuth attackers due to inadequate domain ownership verification. Consequently, we prioritize NS domains with high *NS-DN* values in the TLD zone files for analysis.

Based on the nameserver seeds, the second module automatically expands and aggregates the NS domains to uncover SharedAuth pools. This process employs a *cross-resolution verification*d method coupled with a *resolution-task reduction* strategy. The underlying concept is that if two NS groups share the same DNS infrastructure, one NS group can resolve the domains delegated to the other group. Accordingly, XDAuthChecker merges these two NS groups if they successfully support cross-resolution for all delegated domain names.

In the third module, we identify vulnerable entities (i.e., flawed hosting providers and affected organizations) susceptible to the XDAuth threat model, uncovering nameserver dependencies among these entities. We automatically cluster shared nameservers using multifaceted data, including HTTP fingerprints, WHOIS information, and domain label semantics. This refined clustering procedure helps identify the relevant entities responsible for these shared nameservers. Subsequently, we conduct a manual analysis to identify vulnerable entities. Note that any vulnerable entity within a SharedAuth group can impact all nameservers in that group.



TLD Zone Files
NS RRs
1. Discovering NS Candidates of Hosting Service
NS-DN
Extract <apex-ns> pairs
Obtain NSs with high dependence
NS seeds
2. Uncovering Shared Nameservers
Apex1
Apex2
NS1
NS2
Cross-resolution verification
NS groups
3. Identifying Vulnerable Entities
HTTP
WHOIS
Domain
Entities
Identify NS domain owners
Check domain validation flaws
Vulnerable Entities

Figure 7: XDAuthChecker: Workflow of XDAuth threat measurement.

4.2 Discovering Hosting Service Nameservers

Our detection starts by discovering the NS domains of public hosting providers to uncover SharedAuth pools, as these domains serve as public entry points for accessing and modifying DNS settings in shared nameservers. However, obtaining a complete list of hosting services and collecting their NS domains is challenging, since they are not publicly released. Additionally, due to the lack of ground truth of nameservers for hosting providers, we can not directly link all NS domains in TLD zone files to their respective controlling entities, such as provider names. To this end, we plan to analyze a selection of representative and influential hosting providers based on two criteria: providers that have garnered high research attention or are widely used with a large user base.

Collecting from public reports. For the first criterion, we review previous research papers, blogs, and hosting provider documentation over the past five years [6,20,30,50,60]. Eventually, we manually curate a list of 72 prominent domain hosting providers (shown in Table 3). This process allows us to obtain the NS domains these providers use to delegate customer domain names. These NS domains serve as the seeds for identifying entities with shared nameservers and help us spot services with potential security risks in DOV.

Exploring from zone files. TLD zone files contain all domain delegation information for registered domains from the perspective of domain registries (as shown in Figure 5), which is more extensive than data derived from passive DNS traffic [60]. We introduce the *NS-DN* to measure the popularity of NS domains based on the zone files. This metric reflects the number of apex domains delegated to an NS domain. An NS domain with a high *NS-DN* suggests its adoption by a public hosting service with many customer domains or a large organization with numerous self-managed domains. Such NS domains deserve further investigation for their security risks.

In experiments, we extract the `NS` records for over 200 million apex domains across 1,090 TLDs' zone files (detailed in Section 5.1). Each `NS` record is converted to an $\langle \texttt{apex}, \texttt{ns} \rangle$ pair, where `apex` and `ns` represent a delegated domain and its NS domain. All such pairs enable us to calculate the *NS-DN* for every NS domain in the zone files. In Section 4.3, as we need to apply a cross-resolution verification process for these NS domains, the network request demand grows exponentially with the number of NSs. Therefore, we establish a lower-bound *NS-DN* threshold, taking into account our network probe capacity and ethical considerations regarding active network detection. All NS domains exceeding this limit will be screened out as potential public nameserver candidates. Based on our statistical analysis (as detailed in Section 5.1), we observe that nameservers serving more than 500 domains account for 95.20% of DNS delegation relationships. Lowering the threshold would double the experiment cost (as discussed in Section 4.3) with limited additional insights and numbers of discovered vulnerable domains. Thus, we set the threshold at 500, which effectively encompasses the majority of prevalent hosting services. As a result, we obtain 25,310 NS domains as the nameserver seeds.

4.3 Uncovering Shared Nameservers

Recall that shared nameservers are categorized into those sharing the same IPs (SharedAuth-I&III) and those with different IPs (SharedAuth-II&IV). The same-IP nameservers can be easily obtained by checking NS IPs. So we extend all $\langle \texttt{apex}, \texttt{ns} \rangle$ pairs in the zone files to $\langle \texttt{apex}, \texttt{ns}, \texttt{ip} \rangle$ by querying IP addresses of the `ns` set, and expand the 25,310 nameserver seeds by grouping the `ip` set.

However, identifying shared nameservers with different IPs requires more effort. As per the definition of a nameserver (Section 2.1), we infer that if two nameservers can provide authorized DNS RRs to all domains delegated to each other, they likely belong to the same SharedAuth pool. Additionally, during our empirical verification, we confirmed that unlike the HTTP Host checking in web scenarios, nameservers lack access isolation for differentiating queries toward different NS domains. This is because resolvers directly query the IPs of the nameservers. Consequently, the stored DNS RRs can be retrieved through any NS domains in the SharedAuth pool.

With this in mind, we classify two NS domains as shared nameservers if they share the same IP addresses or provide authoritative responses for domains delegated by the other. Building upon this assumption, we introduce a novel approach called *cross-resolution verification* to identify shared nameservers efficiently. The detailed methodology is as follows:

Cross-resolution verification. For each nameserver seed, we initially choose three delegated domains at random from the TLD zone files and collect relevant $\langle \texttt{apex}, \texttt{ns}, \texttt{ip} \rangle$ pairs. If NS domains are associated with the same `ip`, we directly group them together. Otherwise, we conduct a Cartesian product cross-mapping between the `apex` set and the `ns` set, as the second process in Figure 7. Based on these mappings, we crossly resolve each apex domain by setting the DNS servers as corresponding nameservers. The cross-resolution allows us to verify the capability of the re-mapped NS domains to handle queries for a specific apex domain. If two nameservers simultaneously respond to the same domain queries, we consider them to share a DNS database and are part of SharedAuth.

An exception is that registrants might configure multiple NS domains, managed by different parties, for a single domain to enhance the robustness of authoritative services. These nameservers are not SharedAuth, though they could answer for the domain. Thus, during the random selection of delegated domains, we only use the `apex` set hosted by one entity to minimize false positives resulting from user configurations. For further domain dependency analysis, we empirically treat different SLDs as controlled by different entities.

Resolution-task reduction. Despite we have narrowed down nameserver seeds by setting a lower *NS-DN* bound in Section 4.2, the query volume during cross-resolution remains substantial. Brute force cross-resolution of all `apex` and `ns` domains would escalate network requests to $3n^2$ (with n being the `ns` number). Only 25k nameserver seeds would generate nearly 1.9 billion domain resolution probes, not to mention analyzing the entire TLD zone file dataset, which includes over 2 million NS domains. To address ethical concerns regarding active probing, we employ a pruning strategy to reduce resolution tasks, mitigating network request load and avoiding overburdening the actual DNS infrastructure. Eventually, we effectively reduced the active query volume by 85.94%. The pruning idea is to minimize n as much as possible:

First, for NS domains under the same NS apex domains (SharedAuth-III&IV), we assume they are owned by the same entity and only select one representative. We validate its rationale by conducting a preliminary experiment, where we perform cross-resolution verification within NSs under the same apex domains and divide them into subgroups. According to our dataset, over 97.88% of NS apexes have no more than 10 FQDNs, resulting in a domain query volume of only 2 million for this experiment. Based on the experiment results, we find the average subgroup number under each NS apex (or possibly each service entity) is 1.84, indicating that only 1.84 NS domains are needed per NS apex domain. As a result, we reduce the NS candidate count (i.e., n) by 58.81%.

Second, for NS domains under various apexes but resolving to the same IP address (SharedAuth-I), we select a representative from these as well. This approach stems from the fact that a single IP address typically hosts one DNS server. Therefore, it is unnecessary to repeatedly verify whether such NS domains are part of SharedAuth pools.

Third, we filter out NS domains with unusual resolution behaviors [59]. Our analysis mainly covers two types of unusual behaviors. Some NSs allow recursive resolution for domains not delegated to them, which falls outside our study focus. Some other NSs respond to arbitrary domain queries with fixed IP addresses (e.g., 127.0.0.1), similar to parking services like iPage, which we also exclude from our analysis.

Experiment considerations. First, we minimize domain resolution overhead by randomly selecting a specific NS IP address from multiple addresses that an NS domain adopts for load-balancing purposes. To assess response consistencies among different NS IP hosts, we conducted a preliminary experiment and found that only 3.4% of NS domains exhibit response variations, primarily due to occasional `SERVFAIL` (DNS Return Code: 2) caused by certain IP hosts. To mitigate this variability, we choose an active IP for cross-resolution tasks instead of relying on the NS domain. This approach helps us circumvent the issues of dynamic DNS IP switching.

Second, we conduct scanning tasks from multiple vantage points (Hong Kong, Dubai, and Virginia) to overcome accessibility issues due to censorship or geographical biases [55]. We take a union of the results from different locations, which ensures a broad data collection and can ensure the accuracy of our results. However, this collection only includes nameservers with perfectly consistent responses, representing a lower bound for all shared nameservers.

4.4 Identifying Vulnerable Entities

Next, we identify the vulnerable entities (i.e., providers or organizations) deploying the shared nameservers. The key challenge lies in accurately mapping nameservers to their managing entities, as no public dataset is available for this purpose. Additionally, even if we know a nameserver's owner, verifying its vulnerability requires complex configuration and manual testing. To address this, we first use information of NS domains to identify their owners. Then, we target the top clusters of shared nameservers and enlist volunteers to verify their vulnerability. If anyone within a cluster is vulnerable, the entire group is affected.

Nameserver owner identification. We utilize multifaceted sources, including HTTP content, WHOIS data, and entity names revealed from domain labels, to identify nameserver owners. See Algorithm 1 for the pseudocode. First, we gather HTTP responses and WHOIS data for the shared nameserver's SLD. Then, we extract owner information from the `Title` and `Copyright` sections in HTTP responses, and from the registrant organization in WHOIS records. This yields 1,867 entities from HTTP and 930 from WHOIS. However, entities may utilize nameservers across various TLDs. For instance, both `azure-dns.com` and `azure-dns.net` are owned by Azure. Thus, we further aggregate the above entities based on domain name labels, resulting in 2,403 identified entities.

Cross-entity inspection. Inspecting cross-entity exploitation requires manual setup, such as registering accounts, delegating domains, and checking DOV processes. With thousands of NS groups and relevant entities, we recruit 10 students working in cybersecurity research to assist in testing the entities within the top 200 NS groups. Note that we only test enti-

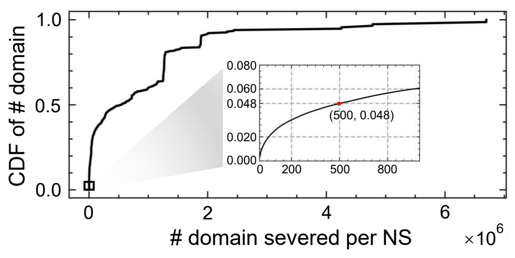


Figure 8: Distribution of delegated domain numbers per NS.

ties offering free or trial public hosting services. For each test group, we prioritize the NS domain serving the most customer domains, which might be a nameserver from providers. For unlabeled NS domains, we manually determine the belonging entities of nameservers using search engines. We conduct *E1* testing for groups with multiple hosting providers and *E2* testing if a group contains only one public provider.

E1: Hosting and hosting. For groups with multiple hosting providers, we register *victim* and *attacker* accounts on two providers (e.g., providers A and B) and conduct the following processes. First, we delegate a test domain using the *victim* account on provider A and then discontinue the service while leaving the NS records unchanged. Second, we claim the test domain using the *attacker* account on provider B and configure a carefully crafted TXT record. Then, we verify whether the domain delegation is successful and check if we can query the TXT record. A successful outcome demonstrates that the attacker has taken over the test domain.

E2: Hosting and non-hosting. If a group contains only one hosting provider, confirming XDAuth threats is challenging since we cannot host domains to the nameservers of non-hosting entities. To this end, we select domains that delegate to non-hosting nameservers and present REFUSED return codes as our test domains. Then, we select two test domains per group and perform cross-entity tests via the hosting provider, i.e., trying to claim the test domain on the hosting provider and set test TXT records. If any test domain is vulnerable, we consider the hosting provider allows cross-nameserver queries without XDAuth protections.

Discussion. For minimizing test influences, we only select test domains with low query volume based on passive DNS data from VirusTotal [52], set TXT records, and promptly remove the records to restore the domain to REFUSED after tests. Additionally, we inform domain owners of our actions and the risks their domains face. Section 6.4 delves into ethical considerations.

5 XDAuth Threat in the Wild

5.1 Dataset Overview

TLD zone files. The input of XDAuthChecker is TLD zone files, which provide comprehensive authoritative DNS records for registered domain names. To explore the real-world domain delegation landscape, we gathered 1,090 authorized zone files through ICANN's Centralized Zone Data Service [29], covering 19 generic TLDs (gTLDs) like .com and .org, along with 1,071 new gTLDs like .site and .xyz.

Our study primarily focuses on stable sharing relationships among nameservers, which are less prone to sudden changes, despite TLD zone files being updated daily. Thus, we limited our analysis to a snapshot from September 13, 2023. This dataset comprises 2,648,840 NS domains, spanning over 200M apex domains. Notably, nameservers serving over 500 domain names accounted for 95.20% of the delegations according to Figure 8. Considering the NS popularity as discussed in Section 4.2, we selected 25,310 NS domains that serve over 500 apex domains as our research objects.

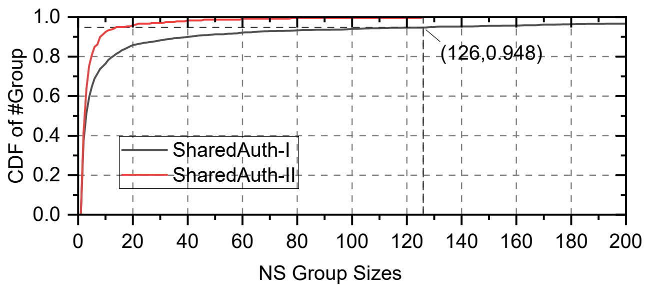


Figure 9: CDF of NS group sizes for SharedAuth-I and SharedAuth-II.

5.2 Landscape of Shared Nameservers

Our cross-resolution experiments confirm the pervasive practice among providers of sharing DNS infrastructure. Below, we will explore the landscape of different SharedAuth types.

Understanding SharedAuth-I and SharedAuth-II. We identified 2,134 SharedAuth-I groups and 238 SharedAuth-II groups, comprising 60,974 and 4,800 nameservers, respectively. Additionally, 1,359 nameservers are involved in both types. The group size distributions of these two types are shown in Figure 9, with SharedAuth-I displaying larger NS group sizes. Approximately 5% of nameservers are shared across more than 126 NS domains. Among these, 16 IPs of Bluehost and HostGator serve over 1,000 NS domains each. Instead, within SharedAuth-II, the max group includes 126 NS domains, but 95% of the NS groups have fewer than 18 NS domains.

A wide variety of entity types tend to share DNS infrastructure. They include prominent DNS hosting providers (e.g., Amazon Route 53 and NSONE), web hosting providers (e.g., Squarespace and NationBuilder), domain registrars (e.g., Godaddy and Hichina), as well as renowned corporations (e.g., Nike and McKesson). In addition, this practice is also seen in some governments (e.g., NJ.gov) and famous media websites (e.g., the BBC).

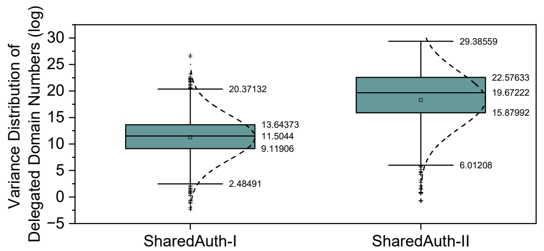


Figure 10: The variance distribution in the delegated domain numbers per NS in each NS group.

Entities within each group may have notable variations in delegated domain numbers and service types. For instance, a hosting provider (e.g., Cloudflare) typically delegates more domain names than a specific enterprise (e.g., McKesson) that establishes nameservers primarily for its own websites. Figure 10 displays the variance distribution of delegated domain numbers per NS domain within each NS group. It highlights that nameservers might be shared by various types of entities with differing popularity. For example, the popular domain registrar Whois.com manages an NS domain, ns1.whois.com, which shares an NS IP with over 1,679 other NS domains. These NS domains belong to registrars like Regway and BigRock, web hosting providers like BanaHosting, and specific businesses like Waycomp Solutions.

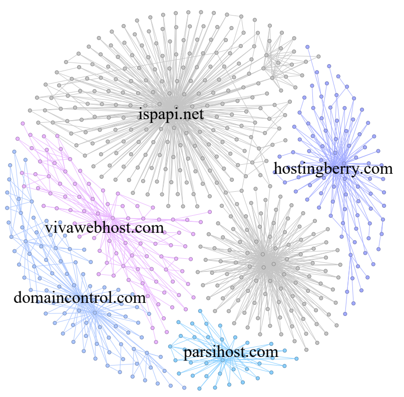
Relationships among entities under SharedAuth-I and SharedAuth-II. In examining the sharing relationships of NS SLDs among different entities, we observed distinct converging clusters formed by the identified NS groups, as depicted in Figure 11. We find that *if any nameserver in an NS group is vulnerable, it could potentially affect all other organizations in that group.* For SharedAuth-I, we showcase the top five NS groups with the most shared nameservers in Figure 11a. The top two prominent groups are `ispapi.net` and `domaincontrol.com`. They are managed by two domain registrars, Hexonet and GoDaddy, respectively. In addition, Figure 11b displays the sharing relationships within the SharedAuth-II category. The two largest groups originate from Bluehost and NSONE[4]. NSONE, in particular, offers authoritative DNS services to a wide range of customers, encompassing web hosting providers like Squarespace, domain management services such as ComLaude, and notable corporations like Porsche and McKesson.

Zooming into the organization level, we investigated the relationships between entities sharing DNS infrastructure, uncovering the motivations behind such deployment. We found that nameserver sharing is often influenced by business practices and partnerships. Overall, we identified four common scenarios (depicted in Appendix B) where different entities tend to share nameservers: (i) *Enterprises subscribe to public DNS hosting services.* Many organizations choose public hosting services when establishing their authoritative nameservers. For example, BBC deploys its nameservers through a public DNS hosting service, NSONE. Its NS domains, ns{#}.bbcdns.net., are pointed to the IP pool controlled by NSONE, as shown in Figure 14a. (ii) *Service providers migrate their DNS infrastructures.* As services need upgrading, providers may change their authoritative nameservers, both in terms of NS domains and IPs. Figure 14b demonstrates how GoDaddy has transitioned its NS domains over time. To maintain service continuity, the old nameserver hosts are integrated into the new ones, ensuring that domains using the previous nameservers remain resolvable. (iii) *Consolidation of DNS infrastructure after company mergers and acquisitions.* A case in Figure 14c is the merger between DigiCert and DNSMadeEasy. Post-acquisition, their DNS hosting services were unified, leading to the consolidation of ns{#}.dnsmadeeasy.com (formerly used by DNSMadeEasy) and ns{#}.digicertdns.com (now employed by DigiCert). (iv) *Different services within a single corporation share a common nameserver pool.* Take the case of 1984.is, which provides both DNS hosting (through ns{#}.1984.is) and web hosting (via ns{#}.1984hosting.is) services. The nameservers of both services utilize the same underlying DNS infrastructure, as shown in Figure 14d.

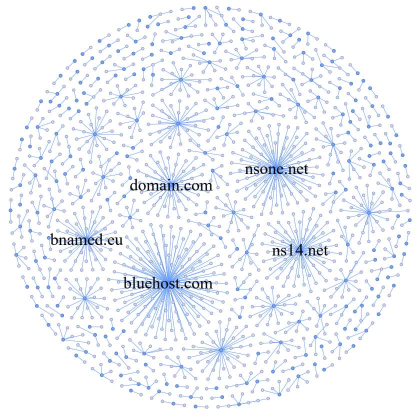
[4]Bluehost is a web hosting provider, and NSONE is a DNS hosting provider.

Understanding SharedAuth-III and SharedAuth-IV. These two categories, encompassing NS domains under the same apex domain, refer to nameservers shared by multiple services within the same providers. Prior research [59,60] has revealed that providers adopt diverse strategies to assign NS domains to different services. To methodically explore these practices, we delved into the SharedAuth pools at the provider level. Following a thorough cross-resolution verification process as discussed in Section 4.3, we identified 10,426 shared groups under these two categories, enabling us to analyze the deployment strategies within a single provider. From these NS groups, we observed the following practices:

First, significant variations exist in NS deployment strategies among different service providers, with some approaches exhibiting vulnerabilities. Figure 12 depicts a comparative representation of each provider's SharedAuth group number against its NS domain number. For instance, GoDaddy (operating NS domains under domaincontrol.com) maintains 94 NS domains and organizes them into 46 nameserver groups. It leverages the separated nameserver groups to serve different user groups. Conversely, Cloudflare manages nearly 1k NS within a single nameserver group but implements an isolation strategy to constrain the operation ability of unauthorized users. A user-configured DNS RR can only be queried from the allocated nameservers if domain ownership validation (i.e., NS checking) fails. Both practices are considered secure in domain delegation validation. However, not all providers employ such secure practices. A case in point is NSONE, which synchronizes user-configured records across all nameservers



(a) SharedAuth-I (Top 5 groups)



(b) SharedAuth-II

Figure 11: Nameserver sharing relationships. Each node symbolizes an apex domain of an NS, with its degree indicating the count of NS apex domains sharing DNS infrastructure with it. A node cluster depicts an identified NS group.

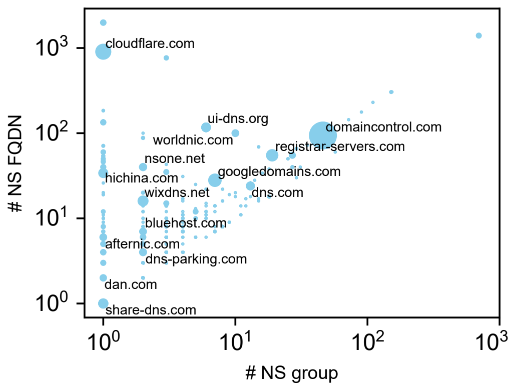


Figure 12: Distribution of each provider's NS FQDN count and SharedAuth group number. The providers are represented by circles whose sizes are proportionate to the number of delegated apex domains.

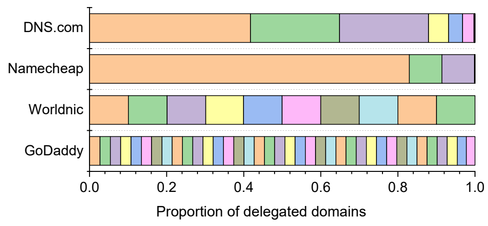


Figure 13: Examples of provider-specific nameserver preferences. The area of each bar segment reflects the proportion of domains served by each NS group.

in its SharedAuth pools regardless of whether the delegation validation is successful or not. This approach potentially creates vulnerabilities, as it allows adversaries an opportunity to inject malicious DNS records into the SharedAuth pools, thereby enabling the hijacking of victim domains.

Second, some service providers have preferences towards particular authoritative NS domains. The preference analysis involves quantifying the domain names managed by each NS group. Figure 13 displays the usage pattern examples of multiple NS groups across providers. While GoDaddy and Worldnic distribute user domains uniformly among their NS groups, providers like DNS.com and NameCheap show a noticeable bias toward the use of NS groups.

Third, despite rare exceptions, most NS domain names corresponding to multiple IPs indeed not only provide load balancing but also offer consistent results. Many providers assign multiple IPs to a single NS domain for load balancing, enhancing service quality. In our study, we collected 4,196 NSs with multiple IPs through multiple measurements at 3 locations and found that 4,055 (96.6%) NSs returned consistent results. For those yielding inconsistent results, our manual analysis attributed these discrepancies primarily to service quality issues or misconfigurations within the NS, often resulting in `TIMEOUT`, `REFUSED`, or `NXDOMAIN` responses.

5.3 Real-world Impact of XDAuth Threats

Overview of vulnerable providers and domain names. As shown in Table 1, our analysis identified 12 public hosting providers vulnerable to XDAuth attacks, all participating in shared NS groups with varying pool sizes, from 5 to 1,386 NS members. These vulnerable providers affect 1,881 authoritative nameservers (termed affected NSs below) of over 67 organizations that share DNS infrastructures with them, like McKesson. Among the affected NSs, 981 are ranked within

Table 1: Identified potentially vulnerable providers with shared nameservers.

Provider	Service Type	Threat	SharedAuth Category	# Shared NS		# Delegated Domain*	
				Total	Vulnerable	Total	Vulnerable
Digicert DNS [17]	DNS Hosting	T1, T2, T3	SharedAuth-I	1,199	850	797,782	50,546
Anon.C	DNS Hosting	T1, T3	SharedAuth-I&II	19	8	5,632,946	27,277
Amazon Route 53 [7]	DNS Hosting	T1, T4	SharedAuth-I	1,386	601	196,161	22,128
NSONE [43]	DNS Hosting	T1, T4	SharedAuth-I	66	45	362,302	12,211
Huaweicloud DNS [28]	DNS Hosting	T4	SharedAuth-II	35	8	330,933	6,962
DNSimple [19]	DNS Hosting	T1, T4	SharedAuth-I	45	26	185,971	3,194
Artglider [10]	Web Hosting	T1, T3	SharedAuth-I	776	248	25,954	1,660
1984.is [1]	DNS & Web Hosting	T1, T3, T4	SharedAuth-II	5	4	25,498	378
Hosting.de [27]	DNS & Web Hosting	T1, T3	SharedAuth-I	134	29	13,288	369
PointDNS [46]	DNS Hosting	T1, T4	SharedAuth-I	11	9	3,016	181
Exclusive Hosting [22]	Web Hosting	T1, T4	SharedAuth-I	185	45	5,447	165
Infinityfree [31]	Web Hosting	T1, T3	SharedAuth-I	71	8	10,053	53

*: The number of domain names delegated by shared NSs.

the Tranco[5] top 1M. Such vulnerable providers enable attackers to hijack domains delegated to the nameservers in these groups, like Canon, Porsche, and the BBC. Notably, DigiCert DNS and Amazon Route 53 emerge as the most critical providers, each potentially compromising the security of over 500 nameservers in their respective sharing pools.

It is evident that most vulnerable public hosting providers fall into the category of SharedAuth-I, referring to Table 1. This means the providers typically share DNS infrastructure by directing their NS domains to identical IP addresses, often managed by major hosting providers. Meanwhile, three providers are involved in SharedAuth-II. In such cases, even though NS domains and NS IPs differ, these systems likely rely on a shared backend DNS storage or cache system.

Additionally, to evaluate the influence caused by these providers, we extracted all apex domains delegated to the 1,881 affected nameservers from the TLD zone files and conducted testing as introduced in Section 4.4. Through this process, we pinpointed 125,124 domains vulnerable to XDAuth attacks. These vulnerable domains are hosted on the affected NSs associated with various types of entities, including web hosting services (e.g., Squarespace), domain registrars (e.g., Bigwww.com), and major organizations like Porsche, the World Bank, and the BBC.

Lessons from the vulnerable providers. First, most (10 of 12) of the vulnerable providers do not deploy any measures against domain name hijacking, allowing customers to claim any unauthorized domain names. Additionally, their NS allocation strategies are also vulnerable, enabling attackers to easily allocate the same NS set with victim domains. Second, certain providers (e.g., *Anon.C*) have implemented domain ownership verification, but they only check subdomains, neglecting apex domains. As a result, they fail to fail to prevent XDAuth. Third, some providers like Amazon Route 53 have made significant efforts to counter domain name hijacking [50]. They monitor all domain delegation pairs and do not allow any new zones of the domain to be created on those name servers. While this approach effectively mitigates previous domain takeover attacks, XDAuth's exploitation point lies in the shared DNS infrastructure rather than the conventional delegation relationship, which circumvents the delegation set inspection of Amazon Route 53.

[5]Tranco is a research-oriented top site ranking list that mainly consists of apex domains. https://tranco-list.eu/methodology

Threat surfaces of XDAuth. The XDAuth threat spans a wide range of enterprises and organizations, leading to four distinct threat surfaces in cross-provider scenarios.

T1: Hijacking privately controlled corporate domains. Traditionally, hijacking domains controlled by a corporation or an organization is highly challenging without compromising the nameservers themselves. XDAuth, however, breaks down the boundaries of authoritative nameservers by exploiting SharedAuth pools. It allows for the injection of crafted records into shared DNS RR storage, paving the way for hijacking privately controlled domains.

T2: Hijacking domain names of legacy customers of hosting providers. Despite certain NS domains no longer being assigned to customers, XDAuth attackers can reactivate the resolution paths of these historical NS domains via new ones, making legacy customers' domains vulnerable to hijacking.

T3: Hijacking domain names delegated by domain registrars. Authoritative services provided by registrars are typically secure, as many do not offer public DNS hosting services. Although some provide it, they can directly use domain registration data for ownership validation. However, XDAuth targets vulnerabilities in DNS hosting providers sharing nameservers with registrars, allowing attackers to circumvent domain ownership validation and facilitate domain hijacking.

T4: Hijacking domains delegated to web hosting providers. This is the most typical scenario where two hosting providers adopt SharedAuth. Attackers can manipulate the storage of SharedAuth pools to activate hidden resolution paths, covertly hijacking customer domains from other web hosting providers.

Table 2: Examples of indirectly affected businesses.

Provider[1]	AffectedBiz[2]	# Domain[3]	Business Field
NSONE	Squarespace	7,613	Web Hosting
	Com Laude	2,025	Domain Management
	BBC	137	News and Media
	McKesson[4]	80	Health
	Porsche	77	Vehicle
DigiCert DNS	Zacco Digital Trust	775	Cyber Security
	McKesson[4]	58	Health and Wellness
	Canon	37	Optics& Image
Anon.C	Bigwww.com	1,346	Registrar
Amazon Route 53	Ezoic	244	Digital Solutions
	Nike	66	Shopping
	Anon.E	57	Entertainment
DNSimple	Sectigo	184	Certificate Management
PointDNS	BrightFire	45	Digital Marketing

[1]: Examples of vulnerable providers.
[2]: Affected businesses or organizations.
[3]: Numbers of vulnerable domains.
[4]: Of McKesson's six NSs, four are shared with NSONE and the remaining two with DigicertDNS. We find all these NSs vulnerable.

Case Studies. We conducted a detailed analysis of the organizations associated with the affected NSs, highlighting examples from diverse industry fields in Table 2.

Case 1: Sectigo is the leading provider of automated certificate lifecycle management and digital certificates. It shared nameservers with DNSimple, putting 184 of its domains at risk of hijacking, which could affect the security of its customers' certificates.

Case 2: McKesson presents another notable example. It utilizes six authoritative nameservers, four of which are shared with NSONE and two with DigiCert DNS, likely as a redundancy measure for service continuity. However, both NSONE and DigiCert DNS's susceptibility to XDAuth attacks leaves 138 of McKesson's domains vulnerable to hijacking.

Case 3: Bigwww.com, functioning as both a domain registrar and DNS hosting provider, has established protective policies based on domain registration information to prevent domain takeovers. However, its shared DNS infrastructure with *Anon.C* opens a backdoor for XDAuth attackers to compromise its customer domains.

6 Discussion

6.1 Lessons Learned

First, securing the DNS infrastructure requires collaboration among various stakeholders, including registries, registrars, registrants, and DNS hosting platforms. With the development of the Internet, DNS is becoming more and more complex, with centralized infrastructure [40, 57], opaque public DNS resolvers with varying behaviours [48], and diverse deployment models [26], which presents both opportunities and challenges. The emergence of new services and features will enhance DNS's robustness and improve the user experience. However, it will also introduce potential attack surfaces. For example, MaginotDNS [34] exploits the resource record isolation vulnerabilities of the resolver in both forwarding and recursive modes to implement cache poisoning. Moreover, XDAuth reveals that cloud hosting providers fail to isolate data in customized DNS zones properly, enabling different NS to manipulate the same DNS RR database.

Second, emerging DNS hosting services are better to adhere to the best security practices of well-managed service providers. Successful examples include Godaddy and Cloudflare, which effectively thwart XDAuth attacks. GoDaddy employs a randomization strategy to prevent attackers from obtaining the same NS set as the victim's domain name, while Cloudflare isolates customers' domain names to prevent unauthorized claims. These practices prevent attackers from tampering with resource records stored in shared nameservers, thereby enhancing security.

6.2 Mitigation

In light of the potential trade-offs between availability and security, and following the analysis of well-managed providers, we provide a series of effective mitigation suggestions for hosting providers and domain owners:

Improve existing NS allocation strategy. NS allocation strategy is an effective policy for mitigating domain takeover threats on DNS hosting platforms. We suggest all providers refer to the secure practices adopted by well-managed providers like Godaddy and Cloudflare as discussed in Section 6.1.

Implementing discontinuation constraints upon customer service termination. A root cause of domain takeover is the failure of registrants to update the delegation information of their domain names after terminating services. While service providers cannot directly manage the NS records for customers' domain names, they can require customers to update NS records upon service termination and proceed with user operations only after confirming the modification.

Maintaining a global status of domain hosting. While Amazon Route 53 has implemented significant measures to thwart domain takeover, it remains susceptible to XDAuth. This vulnerability stems from oversights in managing shared nameservers, creating gaps in their DOV policies. Therefore, providers must maintain a comprehensive and up-to-date overview of global domain delegation statuses within shared nameserver pools. This approach is vital to prevent security breaches that might arise due to discrepancies in the hosting status across various service providers.

Employing domain registration information to perform domain ownership verification. A major challenge in verifying domain ownership is the difficulty in accessing information, such as incomplete WHOIS. The domain registration information held by registrars can serve as effective supplementary data. Thus, hosting service providers can use the registration information from registrars to verify the ownership of domains, thereby alleviating XDAuth. *Anon.C* has adopted this suggestion and is in the process of deploying this solution.

Moreover, domain owners should also make efforts to mitigate the risk. All stale records should be promptly purged when they are no longer in use. This is particularly critical for large companies with global subsidiaries and hosting service vendors, as they may overlook DNS settings for the large volume of subdomains. Our disclosure brought this issue to *Anon.E*'s attention, prompting them to conduct an inspection of all their domain names. Additionally, well-configured DNSSEC ensures the authenticity of DNS RRs using digital signatures, which could effectively mitigate domain hijacking attacks, including XDAuth. DNSSEC records need to be actively generated and configured by the registrant, and not only in their nameservers but also in the parent zone (i.e., TLD).

6.3 Responsible disclosure

We responsibly disclosed to the affected service providers and corporations in compliance with vulnerability disclosure principles [24, 54]. Both *Anon.C* and Amazon Route 53 have acknowledged this vulnerability. *Anon.C* is currently implementing our proposed mitigation solution, while we are actively working with Amazon Route 53 to develop appropriate countermeasures. Also, NSONE has fixed the issue we reported. Furthermore, among the indirectly affected enterprises, *Anon.E* not only acknowledged the threat but also classified it as a CVE-level vulnerability, leading to the prompt fix of the hijackable domains. We noticed that the nameservers for the affected domains were transferred from Amazon Route 53 to CSC's services. In addition, McKesson has confirmed the issue and is working on a fix. We still await responses from other providers. Besides, all responded providers confirmed our testing did not affect their business.

6.4 Ethical considerations

Our experiments were conducted with strict adherence to ethical considerations. These were guided by the principles of the Menlo Report [32] and established best practices for network measurement [44]. Furthermore, our tests adhered to ethical vulnerability disclosure practices as outlined in [25], helping multiple entities mitigate potential threats.

First, we collected and utilized publicly accessible data that does not involve personal information, like TLD zone files, the Tranco domain ranking list, and data from search engines. We strictly followed the recommended practices for using public datasets as outlined in [5].

Second, our cross-resolution experiments required a large number of active queries to target authoritative servers. To mitigate this, we developed an efficient pruning algorithm that utilizes the characteristics of nameservers to reduce the query volume. Meanwhile, we fragmented the tasks across multiple vantage points and limited the query rate from each point to individual NS servers. These measures facilitate reducing the burden on the DNS ecosystem and avoiding excessive network load within a short timeframe.

Third, our cross-entity inspections need to validate the vulnerabilities of affected domain names. In this process, we strived to minimize the actual impact on the hosting providers and the affected organizations. To achieve this, we have designed two different test plans (*E1* and *E2*) based on the scenario, to minimize the potential impact on corporate domains. Moreover, we have also taken several measures to mitigate impact during tests: 1) We target inactive domains and release them immediately after testing, completing the entire process within 2 minutes to prevent client access during our tests. 2) We select a harmless and informational `TXT` as the verification record because multiple `TXT` records can coexist under one domain without overwriting each other, per DNS standards. This ensures our test record does not affect ongoing operations, even if the domain is actively used internally. Furthermore, to mitigate the potential negative cache impact of `NXDOMAIN` on the client, we set the negative cache's TTL to 1 second via SOA records, according to the requirements in RFC1034 [39]. 3) Post-testing, we delete our configured records immediately and verify that the domain has returned to the `REFUSED` status. And, we have checked query logs on hosting platforms and did not observe any requests from normal users during our test. In addition, we make efforts to contact all affected vendors, conduct responsible disclosures, and propose mitigation strategies to help them fix the threat. In discussions, they confirmed our testing did not affect their business. We will continue to engage with affected entities and the security community to promote remediation.

7 Conclusion

In this paper, we perform the first in-depth exploration of the security landscape of DNS hosting infrastructure. Our systematic analysis reveals flawed practices: the service provider's DNS infrastructures lack basic awareness of zone isolation. Based on this, we unveil XDAuth, a novel threat model that leverages shared nameserver infrastructures to breach authoritative service barriers, enabling covert out-delegation domain hijacking. Our findings show that 12 major hosting service providers, including Amazon Route 53, NSONE, and DigiCert DNS, employ inappropriate practices and are exploitable for XDAuth attacks. Through extensive measurements, we demonstrate that XDAuth poses a significant security threat, compromising 1,881 vulnerable nameservers and impacting 125,124 domain names across well-known organizations like Canon, and the BBC. Moreover, we responsibly disclose the issue to the affected providers and corporations. and offer feasible suggestions, and have received acknowledgments from 6 vendors, e.g., Amazon.

Acknowledgments

We sincerely thank all anonymous reviewers and our shepherd for their valuable comments on improving the paper. We also thank Christian Severt of Amazon Security for his support and discussion of our disclosure. This work is in part supported by the National Key Research and Development Program of China (No. 2023YFB3105600), the National Natural Science Foundation of China (62102218), CCF-Tencent Rhino-Bird Young Faculty Open Research Fund (CCF-Tencent RAGR20230116). Haixin Duan is supported by the Taishan Scholars Program. Most of this work was done by Yunyi Zhang during the joint Ph.D. program between National University of Defense Technology and Tsinghua University.

References

[1] 1984.is. 1984-Safe hosting in Iceland. https://1984.is/, 2023.

[2] 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.

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

[4] Gautam Akiwate, Raffaele Sommese, Mattijs Jonker, Zakir Durumeric, kc claffy, Geoffrey M. Voelker, and Stefan Savage. Retroactive identification of targeted DNS infrastructure hijacking. In *IMC*, 2022.

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

[6] 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.

[7] Amazon. Amazon Route 53. https://aws.amazon.com/cn/route53/, 2023.

[8] Amazon. Protection from dangling delegation records in route 53. https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/protection-from-dangling-dns.html, 2023.

[9] Amazon Route 53. Configuring white-label name servers. https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/white-label-name-servers.html, 2024.

[10] Artglider. Web hosting for musicians, labels and bands. https://artglider.com/webhosting, 2023.

[11] Benjamin Braun. Investigating DNS hijacking through high frequency measurements. https://escholarship.org/uc/item/8tm5c7r7, 2016.

[12] Kevin Borgolte, Tobias Fiebig, Shuang Hao, Christopher Kruegel, and Giovanni Vigna. Cloud strife: Mitigating the security risks of domain-validated certificates. In *NDSS*, 2018.

[13] Cloudflare. Custom nameservers. https://developers.cloudflare.com/dns/additional-options/custom-nameservers/, 2024.

[14] ClouDNS. Dns branding / vanity name servers / white-label dns. https://www.cloudns.net/wiki/article/39/, 2024.

[15] Dave Crocker, Tony Hansen, and Murray S. Kucherawy. Domainkeys identified mail (DKIM) signatures. *RFC 4871*, 2011.

[16] Stefan Czybik, Micha Horlboge, and Konrad Rieck. Lazy gatekeepers: A large-scale study on SPF configuration in the wild. In *IMC*, 2023.

[17] DigiCert DNS. DigiCert DNS Trust Manager. https://www.digicert.com/cn/dns-trust-manager, 2023.

[18] DigitalOcean. How to create vanity or branded nameservers with digitalocean cloud servers. https://www.digitalocean.com/community/tutorials/how-to-create-vanity-or-branded-nameservers-with-digitalocean-cloud-servers, 2024.

[19] DNSimple. Dns hosting provider and domain registrar. https://dnsimple.com/, 2023.

[20] EdOverflow. Can i take over xyz? https://github.com/EdOverflow/can-i-take-over-xyz, 2023.

[21] Robert Elz and Randy Bush. Clarifications to the DNS Specification. RFC 2181, 1997.

[22] Exclusive Hosting. Cloud hosting services with a 30-day free trial. https://www.exclusivehosting.net/, 2023.

[23] Google. Verify your domain with a txt record. https://support.google.com/a/answer/183895?hl=en, 2024.

[24] HackerOne. Why you need responsible disclosure and how to get started. https://www.hackerone.com/knowledge-center/why-you-need-responsible-disclosure-and-how-get-started, 2023.

[25] HackerOne. Why you need responsible disclosure and how to get started. https://www.hackerone.com/knowledge-center/why-you-need-responsible-disclosure-and-how-get-started, 2023.

[26] Shuai Hao, Haining Wang, Angelos Stavrou, and Evgenia Smirni. On the DNS deployment of modern web services. In *ICNP*, 2015.

[27] Hosting.de. Webhosting, domains nextcloud aus deutschland. https://www.hosting.de/, 2023.

[28] Huawei Cloud. Huawei cloud DNS. https://support.huaweicloud.com/function-dns/index.html, 2023.

[29] ICANN. The centralized zone data service. https://czds.icann.org/home, 2023.

[30] 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.

[31] InfinityFree. Free web hosting with php and mysql. https://www.infinityfree.com/, 2023.

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

[33] Murray S. Kucherawy and Elizabeth D. Zwicky. Domain-based message authentication, reporting, and conformance (DMARC). *RFC 7849*, 2015.

[34] Xiang Li, Chaoyi Lu, Baojun Liu, Qifan Zhang, Zhou Li, Haixin Duan, and Qi Li. The Maginot Line: Attacking the Boundary of DNS Caching Protection. In *USENIX Security Symposium*, 2023.

[35] Daiping Liu, Shuai Hao, and Haining Wang. All your DNS records point to us: Understanding the security threats of dangling DNS records. In *CCS*, 2016.

[36] Daiping Liu, Zhou Li, Kun Du, Haining Wang, Baojun Liu, and Hai-Xin Duan. Don't let one rotten apple spoil the whole barrel: Towards automated detection of shadowed domains. In *CCS*, 2017.

[37] Maciej Mionskowski. Dangling danger: Route53's flawed dangling ns record protection. https://www.form3.tech/blog/engineering/dangling-danger, 2023.

[38] Matthew Bryant. The .io error – taking control of all .io domains with a targeted registration. https://thehackerblog.com/the-io-error-taking-control-of-all-io-domains-with-a-targeted-registration/, 2017.

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

[40] 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.

[41] Muks Hirani, Sarah Jones, and Ben Read. Global DNS hijacking campaign: DNS record manipulation at scale. https://www.mandiant.com/resources/globaldns-hijacking-campaign-dns-record-manipulation-at-scale, 2019.

[42] Namecheap. Update on recent hosting breach. https://www.namecheap.com/blog/update-recent-hosting-breach/, 2024.

[43] NS1. https://ns1.com/, 2023.

[44] Craig Partridge and Mark Allman. Ethical considerations in network measurement papers. *Communications of the ACM*, 2016.

[45] Victor Le Pochat, Tom van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *NDSS*, 2019.

[46] PointDNS. Pointdns - easy-to-use, reliable, powerful affordable dns hosting. https://pointhq.com/, 2023.

[47] PowerDNS Hosting. Setting up your vanitynamservers with powerdns hosting. https://www.powerdns.net/en/faq/vanity.aspx, 2024.

[48] Audrey Randall, Enze Liu, Gautam Akiwate, Ramakrishna Padmanabhan, Geoffrey M. Voelker, Stefan Savage, and Aaron Schulman. Trufflehunter: Cache snooping rare domains at large public DNS resolvers. In *IMC*, 2020.

[49] Shir Tamari. DNS loophole makes nation-state level spying as easy as registering a domain. https://portswigger.net/daily-swig/black-hat-briefings-hosted-dns-configuration-flaws-risk-leaking-corporate-network-topologies, 2021.

[50] Shiv Sahni. AWS NS Takeover. https://shivsahni2.medium.com/aws-ns-takeover-356d2a293bca, 2019.

[51] Marco Squarcina, Mauro Tempesta, Lorenzo Veronese, Stefano Calzavara, and Matteo Maffei. Can I take your subdomain? exploring same-site attacks in the modern web. In *USENIX Security Symposium*, 2021.

[52] VirusTotal. https://www.virustotal.com/, 2023.

[53] Thomas Vissers, Timothy Barron, Tom van Goethem, Wouter Joosen, and Nick Nikiforakis. The wolf of name street: Hijacking domains through their nameservers. In *CCS*, 2017.

[54] Thomas Walshe and Andrew C. Simpson. Coordinated vulnerability disclosure programme effectiveness: Issues and recommendations. *Comput. Secur.*, 2022.

[55] Gerry Wan, Liz Izhikevich, David Adrian, Katsunari Yoshioka, Ralph Holz, Christian Rossow, and Zakir Durumeric. On the origin of scanning: The impact of location on internet-wide scans. In *IMC*, 2020.

[56] Warren Mercer and Paul Rascagneres. DNSpionage Campaign Targets Middle East. https://blog.talosintelligence.com/2018/11/dnspionage-campaigntargets-middle-east.html, 2018.

[57] Chengxi Xu, Yunyi Zhang, Fan Shi, Hong Shan, Bingyang Guo, Yuwei Li, and Pengfei Xue. Measuring the centrality of dns infrastructure in the wild. *Applied Sciences*, 2023.

[58] Fenglu Zhang, Baojun Liu, Eihal Alowaisheq, Jianjun Chen, Chaoyi Lu, Linjian Song, Yong Ma, Ying Liu, Haixin Duan, and Min Yang. Silence is not golden: Disrupting the load balancing of authoritative DNS servers. In *CCS*, 2023.

[59] 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.

[60] 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. In *SIGMETRICS*, 2023.

[61] 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 *USENIX Security Symposium*, 2024.

A Hosting Service Provider Survey

To provide foundational support for our analysis, we conducted a review of research, blogs, and hosting provider documentation spanning the last five years [6, 20, 30, 50, 60]. Following this, we meticulously collected a list of providers, which is presented in Table 3. This is so far the most comprehensive list showing the NS domain patterns of public hosting services, which is an important seed for this study.

B Share Nameservers Common Scenarios

We investigate the relationships between organizations that share DNS infrastructure, uncovering the underlying motives for such arrangements. Figure 14 shows four common scenarios where different entities tend to share nameservers.

C Nameserver owner identification algorithm

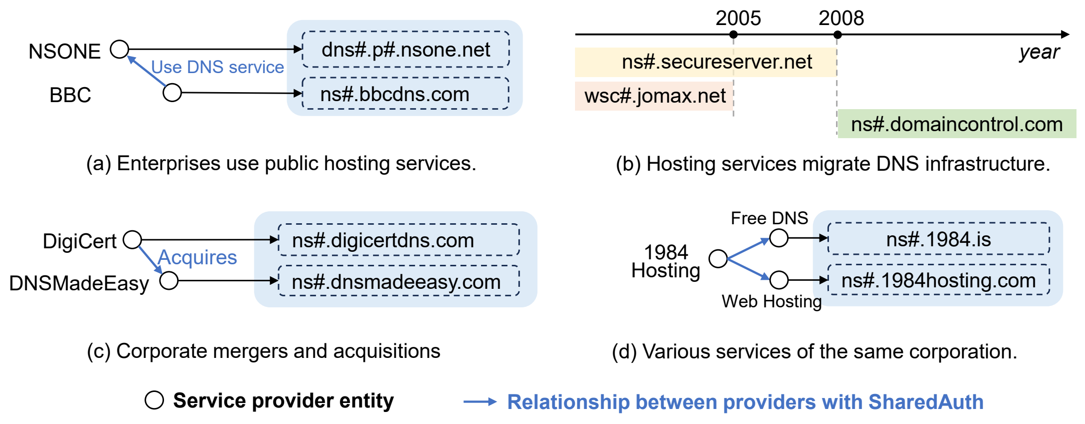
We display the pseudo-code (Algorithm 1) of the nameserver owner identification algorithm.

Algorithm 1 : Identifying nameserver's owner.

```
Input: NS domain ($ns\_list$).
Output: Owner dictionary ($owner\_dict$).
 1: $nsinfo\_list = []$
 2: for each $ni \in ns\_list$ do ▷ Collecting HTTP and WHOIS.
 3:     $hinfo = extract\_http\_response(ni)$
 4:     $winfo = extract\_whois\_info(ni)$
 5:     $ni\_domain = extract\_domain(ni)$
 6:     $nsinfo\_list.append([ni, hinfo, winfo, ni\_domain])$
 7: end for
 8: $owner\_dict = dict()$
 9: $n = len(title\_list)$
10: for each $ni \in nsinfo\_list$ do
11:     if $ni[2]$ not empty then
12:         $owner\_dict[ni] = [ni[2], ni[3]]$
13:     else
14:         $owner\_dict[ni] = [ni[1], ni[3]]$
15:     end if
16: end for
        $owner\_dict = cluster(owner\_dict)$ ▷ Clustering the NS based on same entities and domains.
17: return $owner\_dict$;
```

Table 3: List of service providers surveyed.

Provider	NS allocation	Provider	NS allocation	Provider	NS allocation
DNS.com	ns{1-2}.dns.com	DNSMadeEasy	ns{0-15}.dnsmadeeasy.com	TierraNet	ns{#}.domaindiscover.com
Cloudflare	{#}.ns.cloudflare.com	Bodis	ns{1, 2}.bodis.com	Name.com	{#}.name.com
Godaddy	ns{#}.domaincontrol.com	IONOS	ns{#}.ui-dns.de.; ns{#}.1and1.com	MyDomain	ns{1, 2}.mydomain.com
Gname	{a,b}.share-dns.com	Lolipop (GMO)	uns{01, 02}.lolipop.jp	Linode	ns{1, 2}.linode.com
Network Solutions	ns{#}.worldnic.com	Rackspace	ns{1, 2}.rackspace.com	EasyDNS	dns{#}.easydns.{com, info, org, net}
Siteground	ns{1, 2}.siteground.net	DNSLA	v1s{1, 2}.xundns.com	Dotster	ns{1-2}.dotster.com; ns{1-4}.nameresolve.com
West.cn	ns{1-6}.myhostadmin.net	Onamae (GMO)	dns{01, 02}.gmoserver.jp	Crazy Domains	ns{1-2}.dnspackage.com
Hostinger	ns{1-2}.dns-parking.com	One.com	ns0{1, 2}.one.com	Porkbun.com	{#}.porkbun.com
Xserver	ns{1-5}.xserver.jp	DYN	ns{#}.p{#}.dynect.net	Bigwww.com	ns{11-16}.bigwww.com
Dreamhost	ns{1-3}.dreamhost.com	Netregistry	ns{1-3}.netregistry.net; ns{1, 2}.server-cpanel.com	AliDNS	ns{1, 2}.alidns.com
Hostgator	ns{#}.websitewelcome.com	Value-domain	ns{1,13}.value-domain.com	Amazon Route 53	ns-{#}.awsdns-{#}.{org, com, net, co.uk}
Dynadot	ns{1-2}.dyna-ns.net	SAKURA	ns{1, 2}.dns.ne.jp	Domain.com	ns{1, 2}.domain.com
Donweb	ns{1, 2}.donweb.{cl, bo}; ns3.hostmar.com	Pavietnam	ns{1, 2}.pavietnam.vn	Huaweicloud	ns1.huaweicloud-dns.{com, org, cn, net}
Googledomains	ns-cloud-{#}.googledomains.com	Bizcn	ns{5-8}.cnmsn.net	Baiduyun	ns{1-6}.bdydns.com
Namecheap	dns{#}.registrar-servers.com; freedns{1-5}.registrar-servers.com; {#}.namecheaphosting.com	Hurricane Electric	ns{1-5}.he.com	dnspod	{#}.dnspod.net
Media Temple (GoDaddy)	ns{1-2}.mediatemple.net	GeoScaling	ns{#}.geoscaling.com	gcorelabs.com	ns1.gcorelabs.net
Tucows	dns{1-5}.name-services.com	DNSimple	ns{1-4}.dnsimple.com	000Domains	ns{1, 2}.000domains.com; fwns{1-2}.000domains.com
Hawkhost.com	ns{1-14}.hawkhost.com	FreeDNS	ns{1-7}.afraid.org	Bizland	ns{1,2}.bizland.com; {clickme,clickme2}.click2site.com
A2hosting.com	ns{1-4}.a2hosting.com	1984	ns{0-2}.1984.is	Gandi.net	{a, b, c}.dns.gandi.net
Wix	ns{0-15}.wixdns.net	ClouDNS	ns{#}.cloudns.net	Hover	ns{1, 2}.hover.com
OVH	ns{#}.ovh.net	Contabo	ns{1-3}.contabo.net	NS1	dns{1-4}.p{#}.nsone.net
NIC.ru	ns{#}.nic.ru	Azure (Microsoft)	ns{#}.azure-dns.com	TierraNet	ns{1-2}.domaindiscover.com
Papaki	dns{1, 2}.papaki.gr	Digital Ocean	ns{1-5}.geoscaling.co.	UltraDNS	{p, s,u}dns{#}.ultradns.com
peoplebrowsr.com	ns.peoplebrowsr.com	Reg.ru	ns{1, 2}.reg.ru	Yahoo Small Business	yns{1-2}.yahoo.com



(a) Enterprises use public hosting services.

(b) Hosting services migrate DNS infrastructure.

(c) Corporate mergers and acquisitions

(d) Various services of the same corporation.

Figure 14: Four relationships among parties that have shared nameservers.