Understanding and Exploiting DNS Relaying: Harnessing Legitimate Services for DNS Attacks

Ruian Duan, Shu Wang, Daiping Liu, Hongya Xing, Lexuan Sun,
Yuwen Dai, Zhemin Su, Mengying Jiang, Fan Fei, and Qi Zhang
Palo Alto Networks, Santa Clara, USA
{rduan, shuwang, dpliu, hxing, lesun, ydai, zsu, mjiang, ffei, mzhang}@paloaltonetworks.com

***Abstract*—We introduce a novel technique, DNS relaying, exploiting the fact that many services (CDNs, web/email servers) perform DNS lookups for arbitrary hostnames from untrusted clients. Consequently, DNS relaying turns legitimate services into intermediaries, relaying attacker-originated messages via DNS requests to (internal) resolvers and, ultimately, to target nameservers. Our investigation reveals the root causes of this behavior across various services, including request routing, domain ownership verification, sender tracking, user experience enhancements, and security inspections. Importantly, we uncover a new response-latency covert channel that enables bi-directional communication over these DNS relays. By exploiting benign services as relays, DNS relaying circumvents modern security perimeters and amplifies the impact of DNS attacks. We demonstrate that DNS relaying enhances three legacy DNS attacks by enabling *relayed DNS tunneling* to bypass edge security, *relayed DNS amplification*, and *relayed DNS exploit* to reach otherwise inaccessible DNS servers and clients. Our Internet-wide measurement shows that relayed DNS tunneling can achieve 1,400 bits/request exfiltration and 6 bits/request infiltration; relayed DNS amplification can reach a 535-times packet amplification factor; and DNS relaying expands the DNS exploit attack surface by 8 times in terms of reachable server IPs. We responsibly disclosed our findings to 15 major vendors, receiving acknowledgments from multiple parties. The most affected vendor, Akamai, specifically recognized DNS relaying as a security threat due to our discovered practical infiltration channel.**

I. Introduction

The Domain Name System (DNS) is one of the fundamental technologies powering the Internet. DNS is a hierarchical and distributed database primarily used for translating domain names to corresponding IP addresses. Internet services rely on DNS to route users to specific service addresses, turning DNS into the entry to the Internet. Due to its essential role in Internet access, DNS has served various purposes, from client-side web browsing and email services to server-side load balancing and service auto-discovery.

Typically, the server-side DNS usage does not pose security risks because it is often configured to resolve a predefined set of known benign domains. However, when user inputs, particularly from untrusted public users, can trigger server-side DNS queries, there could be security implications. This behavior effectively constitutes a form of server-side request forgery (SSRF), yet existing SSRF research and defenses [25], [50], [66], [72] focus only on the HTTP layer, neglecting the DNS-layer consequences. Therefore, the prevalence and security implications of this DNS-based SSRF behavior in real-world services remain largely unexplored.

In this paper, we present the first in-depth and systematic study to shed light on this problem. We reveal a novel technique, DNS relaying, which exploits the fact that many legitimate services perform DNS resolution for hostnames received from untrusted clients, thus they act as a DNS relay. DNS relaying effectively transforms legitimate services into *resolvers with limited capability* (i.e. relay resolvers), allowing attackers to specify arbitrary hostnames for DNS lookup, but not to specify query types or directly receive DNS responses. Remarkably, we identify a novel *response-latency-based covert channel* that enables attackers to infer DNS answers from timing variations, unlocking bi-directional communication through otherwise one-way DNS relays.

Understanding DNS Relaying. Theoretically, DNS relaying can potentially exploit any protocol or Internet service that accepts hostnames from public users. Our initial manual analysis of prevalent Internet protocols [24] indicates that HTTP, TLS, SMTP, and QUIC can be abused. We then quantify the scale of DNS relaying's impact through both domain-based and IP-based measurement studies. In the domain-based measurement, we analyze 1.9 million top domains from Cisco [4] and Tranco [51] datasets, revealing the following abuse potential across different protocols: HTTP (4.2%), HTTPS (2.7%), HTTP/3 (0.5%), TLS (1.1%), SMTP (20.2%), and QUIC (15 domains). In the IP-based measurement, we scan the entire IPv4 space for HTTP, HTTPS, TLS, and SMTP services and show that 15.19 million IPs can be abused as relay resolvers. This represents an 8-fold increase of attack surfaces, compared to the 1.80 million open resolvers reported in recent studies [32], [77]. Our investigation also uncovers diverse root causes of these unsolicited DNS resolutions. For example, CDNs resolve hostnames for routing and domain verification; email services check sender domains for tracking and verification; content-fetching applications resolve input URLs to generate snapshots and previews; and security services inspect provided hostnames to identify potential threats.

Security Implications. By exploiting legitimate services as relays, DNS relaying can bypass modern security perimeters and facilitate various DNS attacks. In this paper, we demonstrate its capability of enhancing three legacy DNS attacks. First, we introduce *relayed DNS tunneling*, which evades tunneling detection systems deployed at enterprise network

edges. Legacy DNS tunneling encodes command-and-control (C2) payloads in DNS traffic [15], [21], [73], which can be inspected and blocked by DNS security solutions [18], [47], [70], [79]. In contrast, relayed DNS tunneling encapsulates DNS payload within other protocols (e.g., HTTP, TLS, SMTP, and QUIC), thereby evading detection efforts. Although relayed DNS requests may not directly return responses to the compromised hosts, we uncover a novel response-latency covert channel that enables reliable data infiltration. Second, *relayed DNS amplification* exploits aggressive lookups by DNS relays when no response is received from authoritative nameservers. As a result, DNS relays can be turned into potent amplifiers; in our study, one such relay can achieve a packet amplification factor of up to 535. Third, we demonstrate *relayed DNS exploit*, revealing that DNS relaying can expose 8 times more DNS servers that are otherwise inaccessible from the public. This exposure leaves these internal DNS servers and clients open to DNS exploits [2], [17], [22], [26], [27], [29], [37], [38], [57], [67], [69], [77], [78].

Disclosure and Mitigation. We responsibly disclosed our findings to 15 major vendors. While initially dismissive, the most affected vendor Akamai acknowledged the risk and initiated internal evaluation after our demonstration of covert bi-directional C2 via DNS latency channels. Google and Amazon also acknowledged the issue and are assessing mitigation. Incapsula classified the behavior as abuse, whereas other vendors either showed limited concern or did not respond. Since DNS relaying stems from legitimate use cases and is likely to persist, we recommend best practices for mitigation. CDN providers, cloud platforms, and web servers should implement robust input sanitization and access controls to prevent abuse as DNS relays. For services where DNS resolutions are essential (e.g., email systems, content fetching applications, and security services), protective DNS mechanisms should be deployed to mitigate abuse. In addition, enterprises and ISPs can adopt full traffic inspection to block advanced attacks enhanced by DNS relays, especially when service providers delay implementing defenses.

In summary, our paper contributes as follows.

- *Uncovering DNS relaying and bi-directional C2 channel.* We show how legitimate services can be turned into DNS relays and uncover a reliable latency-based covert channel that enables data infiltration and two-way C2.
- *Comprehensive measurements of DNS relays and their root causes.* We analyze major protocols (HTTP, TLS, SMTP, QUIC) and quantify DNS relaying across top domains and the IPv4 space, identifying the mechanisms that enable it.
- *Demonstrating enhanced DNS-based attacks.* We build a relayed DNS tunneling tool, measure amplification and DoS potential, and show that DNS relays expand exploit reachability by over 8 times.
- *Responsible disclosure and mitigation.* We disclosed to 15 major vendors; Akamai, Google, and Amazon acknowledged the risks, with Akamai identifying DNS relaying as a security threat due to the covert infiltration channel. We provide mitigation guidance for all stakeholders.

II. Background and Related Work

DNS relaying abuses the server-side requests made by legitimate services to carry out DNS-based attacks. This relates to two main areas of prior work: (1) server-side request forgery (SSRF) and (2) DNS threats and their countermeasures.

A. Server-side Request Forgery

SSRF, a well-known security issue in the OWASP Top 10 [46], allows attackers to use controllable inputs to trigger server-side requests (SSR). Existing SSRF research focuses on the **HTTP layer**, where it is commonly exploited to access internal resources otherwise unreachable from external networks. Prior work has studied various HTTP abuse patterns [50], [66], measured its prevalence [72], and proposed mitigation [25]. In contrast, SSR-related behaviors at the **DNS layer** have received significantly less attention. Some out-of-band application security testing (OAST) tools [54] embed unique domain names in payloads and monitor the resulting DNS queries to confirm exploitation. They use DNS resolution as a covert confirmation channel. Recent work [10] shows that some websites resolve domain names embedded in user-controlled HTTP headers like `Host` or `X-Forwarded-For`, unintentionally enabling DNS-based data exfiltration through trusted websites in enterprise networks. Nevertheless, the prevalence, root causes, and security implications of these unintended DNS resolutions remain unexplored.

B. DNS Threats and Countermeasures

DNS can be abused by attackers in two fundamental ways. First, they can misuse DNS to directly target third parties. For instance, attackers leverage DNS tunneling for covert data exfiltration and C2 communication, or utilize DNS amplification to flood victims with large volumes of traffic. Second, attackers can compromise components within the DNS infrastructure, such as registrars, nameservers, resolvers, forwarders, or clients. We term such attacks as DNS exploits, which include cache poisoning, denial-of-service (DoS), phishing, and remote code execution (RCE).

DNS Tunneling. Given the essential role of DNS for modern network applications and services, enterprise networks typically implement permissive security policies for DNS traffic. Consequently, DNS tunneling abuses these permissive policies to transmit unauthorized data by embedding payloads within DNS queries and responses [15], [21], [28], [41], [73] and even timing-based covert channels [60]. Through such channels, attackers can stealthily exfiltrate information or maintain communication with compromised systems, bypassing conventional firewalls and monitoring mechanisms. DNS tunneling is a well-known issue and its traffic typically shows unique patterns compared to normal traffic. Based on this observation, modern defenses, including specialized detection techniques [34], [43], [48] and commercial solutions [18], [47], [70], [79] deployed at enterprise resolvers or network perimeters, have significantly mitigated the risk, limiting the effectiveness of legacy tunneling methods.

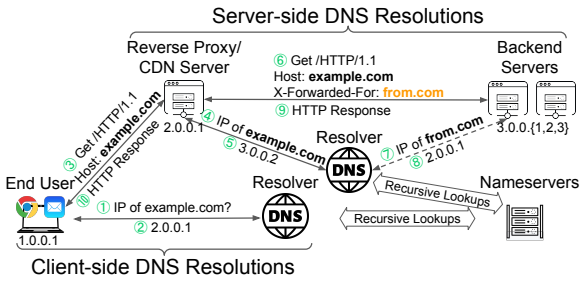


Fig. 1: Examples of legitimate DNS resolutions on the client side and server side.

DNS Amplification. DNS-based distributed denial of service (DDoS) attacks threaten online service availability, by exploiting the disparity between small queries and large responses. Attackers utilize botnets to send forged DNS requests with the victim's spoofed IP address. Because DNS typically operates over UDP, resolvers direct substantial traffic volumes to the victim, amplifying the attack. Vulnerabilities in resolvers, nameservers, and their interactions can further increase amplification [1], [32], [42], [68], [75]. However, DNS amplification's effectiveness has diminished recently, because IP spoofing is increasingly restricted by network filtering [23], [61], and the number of open resolvers is decreasing [30], [31], [49]. Defensive measures, including access control lists (ACL) [45], response rate limiting (RRL) [3], and prompt patching of server vulnerabilities, have further reduced the success rate of DNS amplification attacks [76].

DNS Exploit. Like any network service, the DNS ecosystem is vulnerable to implementation flaws and configuration errors. Recent studies have uncovered various vulnerabilities in resolvers [17], [29], [33], [37], [38], [67], [68], [77], forwarders [31], [78], and DNS clients [2], [22], [26], [57], [69], enabling cache poisoning, DoS, and RCE attacks. However, the practical impact of these vulnerabilities is often limited because many affected servers operate within restricted networks, making them unreachable from the public Internet. Thus, while research continues to identify DNS exploits, their real-world exploitability often relies on network exposure.

III. DNS Relaying

In this section, we introduce DNS relaying, a new class of abuse that exploits server-side DNS resolutions, in contrast to traditional client-driven lookups. We uncover a response-latency covert channel that enables both data infiltration and bi-directional communication through DNS relays.

A. The Hidden Resolution Path

Most prior SSRF research has focused on the HTTP layer, leaving DNS-layer behaviors largely unexplored. To close this gap, we systematically examine unsolicited server-side DNS resolutions. We find that some legitimate services inadvertently forward attacker-supplied DNS queries to internal resolvers and authoritative nameservers, effectively functioning as unintentional DNS proxies. Moreover, their response latency, controlled by DNS resolution times, creates a covert channel through which attackers can retrieve data. We quantify this channel in §VI-A. The essence of the problem is that these hidden resolution paths grant attackers access to internal DNS infrastructure, broadening the attack surface. We refer to this phenomenon as **DNS relaying** and call the affected services **DNS relays** or **relay resolvers**, since they behave like limited-capability resolvers.

To illustrate, Figure 1 contrasts client-side and server-side DNS resolution. On the client side, users browsing the web or checking email trigger resolvers to translate domain names into IP addresses (①, ②). The software then interacts with external services, without visibility into their server-side internals. On the server side, modern architectures such as CDNs and reverse proxies rely heavily on DNS to support routing, caching, and tracking. These components use protocol features such as `Host` (③, ④, ⑤) and `X-Forwarded-For` headers (⑥, ⑦, ⑧) which ultimately require DNS lookups. Server-side DNS resolutions are usually assumed safe because they are configured to resolve a fixed set of trusted domains. Yet when user-controlled input is embedded into these requests, attackers can force servers to resolve attacker domains (e.g., `attacker.tld`). In doing so, the service unknowingly relays queries, enabling the covert latency channel (i.e. ⑩ depends on ⑤, ⑨ depends on ⑧) and establishing bi-directional communication between attacker and nameserver.

DNS relaying allows multiple attack classes to bypass existing defenses. For DNS tunneling, current defenses rely on the inspection of DNS traffic [18], [34], [43], [47], [48], [70], [79]. However, attackers can use DNS relaying to completely bypass these defenses by wrapping tunneling traffic within other protocols, which are then resolved by trusted or widely used services. Unlike malicious endpoints, these intermediary services are legitimate and cannot be simply blacklisted or blocked without disrupting normal operations. For DNS amplification, we find that DNS relays can be exploited to perform repeated lookups, particularly when authoritative servers fail to respond promptly. This behavior allows DNS relays to act as amplifiers. These new amplifiers can be combined with vulnerable resolvers, nameservers, clients, and their interactions to further increase amplification factors [1], [32], [42], [68], [75]. For DNS attacks that exploit resolver vulnerabilities, in addition to timely patching, one of the primary defenses is to use closed or internal resolvers to reduce attack surfaces [17], [29], [37], [38], [67], [77]. The advent of DNS relaying reopens these resolvers to attackers. We discuss the attack details in §VI.

B. Potential Attack Surfaces

DNS relaying exploits the unsafe resolution of untrusted hostnames by legitimate services. In principle, any protocol that is able to include hostnames other than the destination domains is susceptible to these attacks. After systematically reviewing popular protocols on the Internet [24][1], including HTTP, TLS, SSH, FTP, and SMTP, we have identified several vectors. In **HTTP** and **HTTPS** requests, the

[1]The protocols that are deployed on more IPs than DNS (port 53) are selected and reviewed.

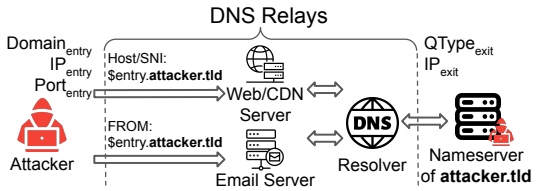


Fig. 2: The scanning client and testing domain setup for both domain-based and IP-based measurement.

`Host` field in the request header is a primary target, along with other header fields such as IP-related header fields (e.g., `X-Forwarded-For`) and proxy-related headers (e.g., `Forwarded`). These header fields, designed for legitimate purposes like virtual hosting or client tracking, can be manipulated to facilitate attacks. The **TLS** Server Name Indication (`SNI`) extension allows clients to specify hostnames during handshake, creating an entry point. In **SMTP** requests, mail servers commonly resolve hostnames supplied in `HELO` and `FROM` fields. Additionally, the emerging **QUIC** protocol inherits the SNI-based risks of TLS, while **HTTP/3**, which runs on top of QUIC, also inherits the header-based risks of HTTP.

IV. Measurement

To enhance comprehension and evaluate the real-world implications of DNS relaying, we conduct measurements to address the following research questions: (1) *How many frequently visited services are susceptible to DNS relaying?* (2) *What is the scale of DNS relaying's impact across the entire Internet?* (3) *Is DNS relaying actively exploited in the wild?* We first describe our measurement setup and then address each question by manually investigating the collected data.

A. Experiment Setup

We evaluate DNS relaying across various popular domain datasets and the entire Internet. From the domain perspective, we select three datasets, namely, *Cisco Top 1M hostnames* [4], *Tranco Top 1M hostnames* and *Top 1M root domains* [51], covering a total of 1.9 million of the most frequently visited hostnames on the Internet. These domain lists were obtained on Jul 23, 2024. From the IP perspective, we scan 8 popular ports across the entire IPv4 address space. Our measurement setup is illustrated in Figure 2. We registered a test domain (denoted as `attacker.tld`) and set up a nameserver to passively receive messages from DNS relays. The nameserver is configured to listen on port 53 without responses, to avoid caching and trigger retries. We denote the public-facing entry domain, IP, and port of a DNS relay as $Domain_{entry}$, IP_{entry}, and $Port_{entry}$, respectively. The IP that connects to our nameserver is denoted as IP_{exit}. The mappings between entries and exits are many-to-many, because a single service may have multiple resolvers to handle resolution failures while multiple services may share the same backend resolver. To map DNS requests received from the exits to their entry points, we encode either $Domain_{entry}$ or IP_{entry} and $Port_{entry}$ into subdomains as `$entry.attacker.tld`. We also record the number of DNS requests received from each relay and their query types as $QType_{exit}$.

1) Top Domain Measurement: We measure DNS relaying across frequently visited (top) domains via various protocols, including HTTP, HTTPS, HTTP/3, TLS, QUIC, and SMTP.

HTTP, HTTPS, and HTTP/3. To understand the feasibility of DNS relaying, we initiate HTTP, HTTPS, and HTTP/3 requests with *customized headers* towards each domain in the top domain dataset. We exemplify this process using the hostname `test.example.com`. We employ the `curl` command to send an HTTP request, with the following headers. HTTPS and HTTP/3 requests use the same header fields, but in different protocols. In a header field, the third-level domain (`$pid`) denotes the specific protocol of this DNS relaying attempt, i.e., `http` for HTTP, `https` for HTTPS, and `http3` for HTTP/3. `$entry` represents a unique identifier assigned to the hostname `test.example.com`. Lastly, the fifth-level domain is an abbreviation of the respective header field. To initiate HTTP/3 requests, we compile `curl` with `wolfSSL`, `ngtcp2`, and `nghttp3` backends [64] to enable QUIC support and use the `--http3-only` flag to enforce the HTTP/3 protocol. Underlying these requests, HTTP is transmitted to target servers over TCP port 80, HTTPS over TCP port 443, and HTTP/3 over UDP port 443.

Template of HTTP Request for Measuring DNS Relays.

```
GET / HTTP/1.1
Host: host.$entry.$pid.attacker.tld
Accept: */*
User-Agent: Mozilla/5.0
Origin: https://test.example.com
X-Forwarded-For: xff.$entry.$pid.attacker.tld
X-Wap-Profile: wafp.$entry.$pid.attacker.tld
CF-Connecting-IP: cfcon.$entry.$pid.attacker.tld
Contact: root@contact.$entry.$pid.attacker.tld
X-Real-IP: rip.$entry.$pid.attacker.tld
True-Client-IP: trip.$entry.$pid.attacker.tld
X-Client-IP: xclip.$entry.$pid.attacker.tld
Forwarded: for=ff.$entry.$pid.attacker.tld
X-Originating-IP: origip.$entry.$pid.attacker.tld
Client-IP: clip.$entry.$pid.attacker.tld
Referer: ref.$entry.$pid.attacker.tld
From: root@from.$entry.$pid.attacker.tld
```

TLS and QUIC. For TLS `SNI`, we utilize the OpenSSL `s_client` tool, a utility for connecting to TLS servers and inspecting TLS certificates. We employ the following command to connect to the TLS server on port 443 for the domain `test.example.com`. The `-servername` option is crucial as it specifies the `SNI` extension, allowing the client to declare the intended hostname during the TLS handshake. `$entry` represents an identifier for the `SNI` value, while `sni` and `tls` are used as protocol-related placeholders.

TLS Request Command for DNS Relay Measurement.

```
openssl s_client -connect test.example.com:443
        -servername sni.$entry.tls.attacker.tld
```

For QUIC `SNI`, given that `OpenSSL`'s QUIC support was experimental, we reuse the same `curl` tool as in the HTTP/3 measurement, but with a customized `SNI`. We first scan the top domains to obtain their IPs and then use the following command to test these domains for DNS relays. For the entry `test.example.com`, `$ip` refers to one of its pre-determined IP address.

The `--resolve` option in the command forces `sni.$entry.quic.attacker.tld:443` to resolve to `$ip`, thereby allowing us to specify the destination as `https://sni.$entry.quic.attacker.tld/`, which will be utilized as the QUIC `SNI`. We also specify the `Host` header field as `test.example.com` to differentiate it from QUIC `SNI` and ensure that `sni.$entry.quic.attacker.tld` is only received from the QUIC layer, rather than HTTP.

QUIC Request Command for DNS Relay Measurement.

```
curl --http3-only -v -k -H 'Host: test.example.com'
   --resolve sni.$entry.quic.attacker.tld:443:$ip
   https://sni.$entry.quic.attacker.tld/
```

SMTP. We initiate our SMTP measurement by scanning the MX records of the top domains and subsequently interacting with the identified mail servers on port 25. To improve scanning efficiency, we customize the *ZGrab* [12] tool to send specific commands to the mail servers. For example, when analyzing the domain `test.example.com`, the following commands are sent to its designated mail server, i.e., `mail.example.com` as indicated in the MX record. Here, `$entry` encodes the connected mail server, while `mail` and `smtp` serve as protocol-related padding. It is worth noting that the number of mail servers is considerably smaller than the number of hostnames, as multiple hostnames often share the same underlying mail infrastructure. For instance, out of the top 1.9 million hostnames, 132K hostnames rely on the Google mail service `aspmx.l.google.com`. Consequently, our analysis only needs to focus on 384K unique mail servers.

SMTP Commands Sent for Measuring DNS Relays.

```
HELO mail.$entry.smtp.attacker.tld
MAIL FROM: <alice@mail.$entry.smtp.attacker.tld>
RCPT TO: <bob@test.example.com>
DATA
.
QUIT
```

2) Internet-wide Measurement: To assess DNS relaying prevalence across the Internet, we conduct a standard two-stage measurement using ZMap [13] and ZGrab [12]: applying ZMap to identify internet-facing IPs with specific open ports, followed by in-depth analysis of each IP with ZGrab.

LZR [24] reveals that ports 80, 8080, 7547, 4567 (for HTTP), 443 (for TLS), 25 (for SMTP), 22 (for SSH), and 21 (for FTP) are commonly open on public IPs and often host both expected and unexpected services. Our manual investigation suggests that SSH on port 22 and FTP on port 21, due to their default authenticated end-to-end communication, are less likely to be exploited as DNS relays. However, LZR reported a significant presence of unexpected TLS and HTTP services on these ports as well. Therefore, we perform simplified DNS relay probes across these 8 ports using ZGrab.

For port 80, we probe HTTP `Host`; for port 443, we probe both HTTPS `Host` and TLS `SNI`; for port 7547, 8080, 4567, 22 and 21, we probe the `Host` field in HTTP and HTTPS, as well as TLS `SNI`. For port 25, we deploy our customized SMTP ZGrab module to send specific commands probing the `HELO` and `FROM` domains. Since SMTP scans require recipient email domains, we scan the open IPs for their PTR records, using the extracted root domains as approximates. Due to the much lower number of DNS relays for HTTP/3 and QUIC (§IV-B1), and the lack of support in ZGrab, we excluded them from our Internet-wide scan. We format the probing hostnames as `$field-$port.$entry.$pid.attacker.tld`, where `$field` represents either `host`, `sni`, or `mail`; while `$entry` and `$port` denote the IP_{entry} and $Port_{entry}$.

3) Vantage Points: We conduct all scans from the United States. For the HTTP and HTTPS top domain measurements, we replicated the experiments across four vantage points to strengthen validity: GCP (US West), Azure (US East), AWS (US East), and a university network in the US East. The results were largely consistent, with only minor variations attributable to CDN providers (§V-B). Based on this consistency, the remaining top domain and IPv4 scans were conducted from a single vantage point, either GCP (US West) or Azure (US East), to minimize traffic and ensure ethical practice.

B. Quantitative Assessment

As summarized in Table I, our evaluation of 1.9 million unique domains reveals that 466K (24.5%) can be abused for DNS relaying. A substantial portion, i.e., 384K domains (20.2%), is attributed to the underlying SMTP mail servers. Besides, 80K (4.2%), 52K (2.7%), and 22K (1.1%) domains are exploitable via HTTP, HTTPS, and TLS, respectively. In IP-based measurements, we identify 15.19 million IPs susceptible to DNS relaying, representing an 8-fold expansion compared to the 1.80 million open resolvers [32], [77]. Port 80 accounts for the majority of DNS relays, contributing 11.36 million (74.8%), while HTTP dominates in protocol-based analysis, responsible for 12.04 million (79.3%) of DNS relays.

1) Top Domain Result: The domain-based measurement in Table I addresses our first research question: *How many frequently visited services are susceptible to DNS relaying*?

HTTP, HTTPS, and HTTP/3. As multiple headers can be exploited for DNS relaying, Table II presents a breakdown of DNS relays by headers for HTTP, HTTPS, and HTTP/3. Among the 1.9 million domains analyzed, 80K (4.2%) can be abused via HTTP, with 76.5K (95.7%) due to improper handling of the `Host` header. Additional attack vectors include 10 other headers, such as `X-Forwarded-For` (3,087 domains, 3.9%) and `X-Real-IP` (258 domains, 0.3%), while other headers affect fewer domains. HTTPS-based DNS relays impact 52K domains (2.7%), again dominated by the `Host` header (46.8K domains, 90.0%), but also introducing new attack surfaces through headers like `From` and `Contact`. HTTP/3 shows the lowest DNS relay rate, affecting 9,776 domains (0.5%), and is excluded from IP-based measurements. Among the vulnerable HTTPS domains, all 12 non-`Host` header fields are exploitable, compared to only 10 in HTTP; `X-Forwarded-For` and `True-Client-IP` emerge as the second and third most prevalent after `Host`, affecting

TABLE I: Summary of domain-based and IP-based DNS relaying measurement for different protocols. Out of popular domains, 24.5% can be abused; compared to open resolvers, 8 times more IP addresses can be abused.

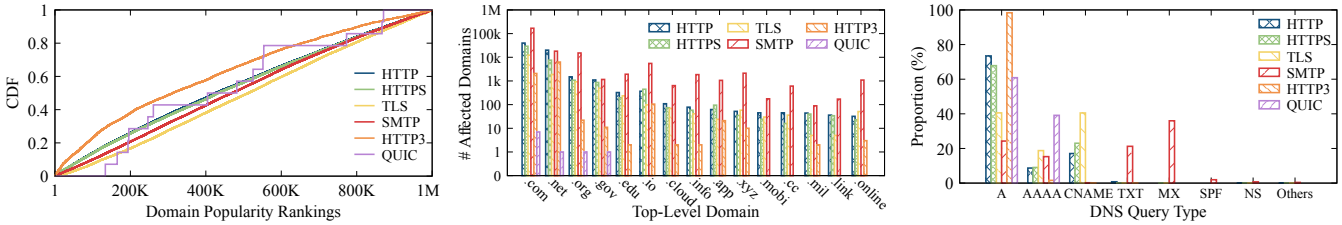
Protocol	# Domains w/ DNS Relays				# IPs w/ DNS Relays								
	Cisco	Tranco(h)*	Tranco(r)	Overall†	80	443	25	7547	8080	4567	22	21	Overall†
HTTP	63,269	39,350	18,339	80,001	11.36M			0.13M	0.45M	0.16M	0.40M	0.15M	12.04M
HTTPS	39,647	27,785	12,655	51,957		2.84M		0.06M	0.07M	0.04M	0.25M	0.16M	3.16M
HTTP/3	9,059	3,818	638	9,776									
TLS	13,536	12,161	10,816	22,110		0.41M		0.11M	0.14M	0.15M	0.28M	0.17M	0.76M
QUIC	1	11	14	15									
SMTP	80,763	271,344	340,038	383,726			1.08M						1.08M
Overall†	136,531	311,440	374,464	466,448	11.36M	3.05M	1.08M	0.17M	0.54M	0.18M	0.68M	0.33M	15.19M‡

* (h): Tranco Top-1M hostname; (r): Tranco Top-1M root domain. ‡: As a baseline, the number of open resolvers is 1.80M [32], [77].
† The number of unique domains and IPs with DNS relays summarized across domain datasets, ports and protocols.

TABLE II: Breakdown of DNS relays by headers for HTTP, HTTPS and HTTP/3 in domain-based measurement.

Header Field in Request	# Domains w/ HTTP DNS Relaying				# Domains w/ HTTPS DNS Relaying				# Domains w/ HTTP/3 DNS Relaying			
	Cisco	Tranco(h)*	Tranco(r)	Overall†	Cisco	Tranco(h)	Tranco(r)	Overall†	Cisco	Tranco(h)	Tranco(r)	Overall†
Host	62,102	37,163	15,870	76,538	37,414	24,513	9,514	46,784	8,928	3,613	495	9,512
X-Forwarded-For	1,059	1,929	2,184	3,087	1,970	2,877	2,756	4,559	118	193	130	238
X-Real-IP	69	183	201	258	57	127	142	196	11	4	4	15
X-Client-IP	31	53	43	69	44	61	48	82	0	0	0	0
True-Client-IP	41	57	58	87	182	115	72	233	0	0	0	0
Client-IP	2	28	22	33	9	42	43	58	0	0	0	0
CF-Connecting-IP	1	1	4	4	4	4	5	8	0	0	0	0
X-Originating-IP	0	0	1	1	0	0	1	1	0	0	0	0
X-Wap-Profile	3	2	3	6	1	5	5	7	0	0	0	0
Forwarded	7	1	2	8	15	5	5	18	2	0	0	2
Referer	3	16	27	31	24	71	98	119	0	8	9	9
From	0	0	0	0	0	3	2	3	0	0	0	0
Contact	0	0	0	0	0	3	2	3	0	0	0	0
Overall†	63,269	39,350	18,339	80,001	39,647	27,785	12,655	51,957	9,059	3,818	638	9,776

* (h): Tranco Top-1M hostname; (r): Tranco Top-1M root domain. † The number of unique domains summarized across domain datasets and headers.



(a) CDFs of domain popularity ranking. (b) # domains grouped by TLDs. (c) $QType_{exit}$ distribution by protocols.

Fig. 3: The domain ranking, TLD, and query type distribution of DNS relaying domains in top domain measurement.

4,559 (8.8%) and 233 (0.4%) HTTPS domains, respectively. In particular, `True-Client-IP`, a CDN-injected header, ranks third in HTTPS while `X-Real-IP`, commonly set by reverse proxies, holds that position in HTTP (258 domains). In contrast, HTTP/3 shows a much narrower attack surface, with only 4 of 12 non-`Host` headers exploitable. Additionally, some domains are susceptible across multiple HTTP headers. Our analysis indicates that in HTTP, 111 domains can resolve hostnames for two or more headers simultaneously. Notably, one domain is affected in 11 distinct HTTP headers, allowing for potential parallel exploitation through a single DNS relay. In HTTPS, 99 domains are capable of resolving hostnames in two or more headers simultaneously, while no domains exhibit this capability in HTTP/3.

TLS, QUIC, and SMTP. Table I reveals that, for TLS, 22.1K out of 1.9 million domains (1.1%) relay DNS requests via TLS `SNI`. This represents a relatively low prevalence compared to HTTP and HTTPS, which have ratios of 4.2% and 2.7%, respectively. Regarding QUIC `SNI`, only 15 domains are found to be affected. Hence, we exclude QUIC from IP-based measurements. Although no definitive conclusions can be drawn, this may be attributed to limited adoption or standardized implementation. For SMTP, 384K domains (20.2%) are affected by DNS relaying, meaning that their mail servers resolve untrusted domains. More specifically, 126K (32.7%) of the 384K mail servers are DNS relays, highlighting the widespread issue of untrusted DNS resolution on SMTP servers. Note that for SMTP relays, the number of popular domains is shown when evaluated against the top domain list, domain rankings, and top-level domains (TLDs). In contrast, when analyzing DNS relays based on IP, query types, and amplification ratios, each mail server domain is treated as a DNS relay, and the number of mail server domains is reported.

Distribution. To characterize what domains are affected, we present CDF of domain distribution against domain popularity in Figure 3(a) and the histogram of their top-level domains (TLDs) in Figure 3(b). Figure 3(a) indicates that DNS relaying is not only observed on lesser-known domains but also on popular ones. For instance, 1,900 domains out of the top 10,000 domains can be exploited as relays. Besides, certain domains with trusted TLDs such as `.gov`, `.edu`, and `.mil` are affected, as shown in Figure 3(b).

2) Internet-wide Result: To address the second research question *What is the scale of DNS relaying's impact across the entire Internet?*, we consider the 1.80M open resolvers reported by recent studies [32], [77] as a baseline. This number reflects a declining trend, as previous studies reported 26.8 million open resolvers in 2015 [30], 13.0 million in 2018 [49]. Building on this baseline, we scan DNS relays across eight ports for HTTP, HTTPS, TLS, and SMTP, identifying 15.19 million potentially exploitable IPs. This represents an 8-fold increase in the attack surfaces compared to the number of open resolvers. From a port-based perspective, port 80 accounts for 11.36 million (74.8%) DNS relays, and the top three ports (80, 443, and 25) collectively contribute 14.31 million (94.2%). The remaining five ports contribute 0.88 million relays, which, although smaller in scale, still represent a significant fraction (48.9%) of the open resolver population. From a protocol-based perspective, HTTP is the most prevalent, accounting for 12.04 million (79.3%) DNS relays. TLS is the least common with 0.76 million, yet still accounts for 42.2% of the open resolver population. A breakdown of the results by port and protocol is provided in Table I.

C. Real-World Exploitation

The OAST technique [53], introduced in 2017, uses controlled DNS queries as evidence of successful exploitation. A 2023 blog post [10] later showed that certain websites resolve hostnames from user-supplied HTTP headers, enabling unintended data exfiltration. Fundamentally, both techniques exploit the core mechanism of DNS relaying by tunneling information from an attacker to a nameserver. Given the growing awareness of these methods, our third research question arises: *Is DNS relaying actively exploited in the wild?* We manually analyzed HTTP(S) session logs from our enterprise customers in April and December 2024, respectively, aiming to understand the trend of DNS relaying abuse. Defining an attack as a unique $(Domain, Customer)$ pair, our analysis reveals a 6-fold increase in attack volume during this eight-month period. Our investigation focuses on the public attack vector: data exfiltration via hostnames (other attacks are detailed in §VI). By analyzing the HTTP(S) session logs (including headers, destination IPs, and anonymized customer information), we identify exploitable headers, e.g., `X-Forwarded-For`, sort them by length, and manually review the longest entries first. Hostnames with encoded data are further analyzed to determine if dedicated nameservers are used to receive information. Table III shows a rise in DNS relaying abuse: malicious domains increased from 5 to 8, and attacks surged from 13 to 82. Domains like oastify.com and oast.fun link to Burp Collaborator [36] and Interactsh [56], while claudfront.net connects to Decoy Dog APT [21]. Notably, queryrecord.com, with nameserver IPs from a public university, shows a significant attack spike in December 2024, impacting numerous customers. These findings highlight the active exploitation of DNS relaying by both security researchers and attackers.

TABLE III: Domains used for data exfiltration and affected enterprise customers in April and December 2024.

Attack Information		# of Customers	
Domain	Attribution	Apr 2024	Dec 2024
claudfront.net	Decoy Dog [21]	2	6
mooo-ng.com	Malicious probing [16]	1	2
oastify.com†	Burp Collaborator [36]	4	5
oast.fun†	Interactsh [56]	3	7
oast.pro†	Interactsh	3	16
oast.site†	Interactsh	0	10
h4.vc†	Interactsh	0	1
queryrecord.com	Public University	0	35
# of Attacks		13	82

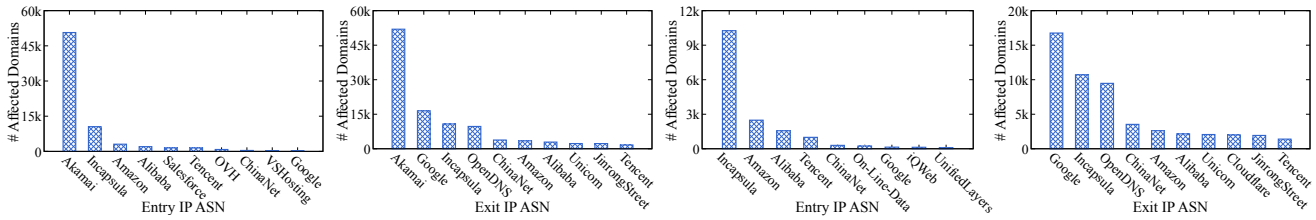
† manually verified as pentest attempts with DNS relaying technique.

V. Root Cause Analysis

To determine the underlying causes of DNS relaying, we begin by mapping IP_{entry} and IP_{exit} to Autonomous System Numbers (ASNs) and combining them with $Domain_{entry}$ to analyze service ownership. By correlating ownership with request details, such as $Port_{entry}$, $QType_{exit}$, we infer the underlying services. To identify and confirm the root causes, we manually examine these services, review their documentation, and perform a best-effort analysis. Our investigation identifies six primary causes of unsolicited DNS lookups: CDN services, cloud providers, web servers, email systems, content fetching applications and security services.

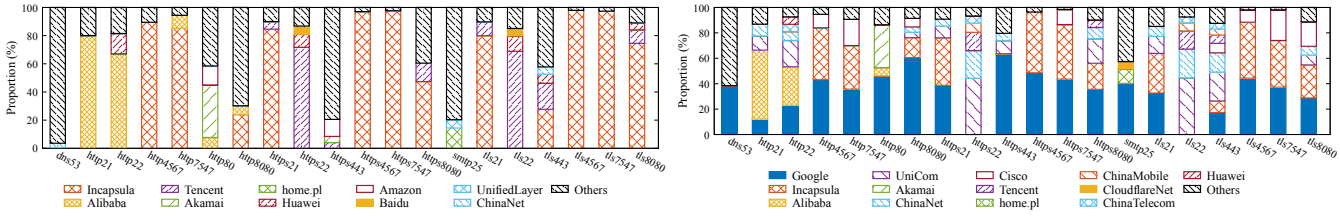
A. Relay Characteristics

From the ASN perspective, Figure 4 shows the top 10 ASNs for IP_{entry} and IP_{exit} in HTTP and TLS relays in the top domain measurement. About 60% of HTTP/HTTPS relays have IP_{entry} from Akamai, and 13–17% from Incapsula, both well-known CDN providers. Incapsula also accounts for 41% of IP_{entry} ASNs in TLS `SNI` relays, while Akamai does not resolve `SNI` extensions. In the Internet-wide measurement, we identify 15.19 million IP_{entry} from 54.9K ASNs and 411.8K IP_{exit} from 16.9K ASNs. Figure 5(a) and 5(b) show the top ASNs for IP_{entry} and IP_{exit} across various protocols and ports. The analysis of IP_{entry} indicates that Akamai, Amazon, Tencent, and Alibaba are the most affected in HTTP (port 80) and HTTPS (port 443) probes. In contrast, Incapsula, Tencent and Alibaba are more commonly observed on other protocols and ports. For IP_{exit}, both domain-based and IP-based measurements show that Google resolvers are commonly used by DNS relays across different protocols and ports, except for HTTPS and TLS probes on port 22. This suggests the widespread adoption of Google resolvers as default or fallback options. Other notable exit resolvers include Akamai, Cisco, Incapsula, Alibaba, ChinaNet, and UniCom. Moreover, we build the mapping between IP_{entry} and IP_{exit} to reveal vendor-specific behavior. For example, Akamai and Alibaba mainly use their own exit resolvers, while Amazon and DigitalOcean prefer Google resolvers. Incapsula relies on a combination of its own, Google, and Cisco OpenDNS resolvers, whereas Tencent utilizes a diverse set including its own infrastructure, UniCom, China Mobile, and ChinaNet.



(a) HTTP (b) HTTP (c) TLS (d) TLS

Fig. 4: Top-10 ASNs of IP_{entry} and IP_{exit} for HTTP and TLS in top domain measurement.



(a) The distribution of IP_{entry} ASN. (b) The distribution of IP_{exit} ASN.

Fig. 5: The distributions of IP_{entry} and IP_{exit} ASNs, over different protocols and ports in Internet-wide measurement.

For DNS requests, the distribution of $QType_{exit}$ across different protocols in the top domain measurement is shown in Figure 3(c). As expected, IP lookups (A, AAAA) are the most common query types across protocols, with Akamai primarily falling into this category. Domain alias queries (CNAME) are prevalent for HTTP, HTTPS, and TLS, and are primarily associated with Incapsula, suggesting distinct root causes for Akamai and Incapsula (§V-B). Mail-related records (TXT, MX, SPF) are exclusively observed from SMTP relays. Similar $QType_{exit}$ distribution patterns are observed in the Internet-wide measurement.

B. Affected Services

CDN Services. CDN services like Akamai and Incapsula serve as intermediaries between clients and web servers, but also represent a significant source of DNS relays. From documentation, we identify two main causes of their unsolicited hostname resolutions. The first is origin routing. For example, Akamai resolves the `Host` header to locate the origin server [65]. This behavior is inconsistent across vendors: DNS relaying works in US West region via Akamai but fails in US East via Fastly, while they both serve the same origin domain `ctldl.windowsupdate.com`. The second is domain verification. Incapsula uses DNS-based CNAME checks to verify domain ownership [20], simplifying onboarding and certificate renewal but exposing additional attack surfaces.

Cloud Providers. We observe that cloud providers like Amazon, Google, Tencent, and Alibaba can resolve HTTP `Host` and TLS `SNI` fields. Upon reviewing their documentation, we found that these providers offer a variety of services, out of which, CDN, load balancers (LB), and web application firewalls (WAF) [5]–[9], [62], may act as DNS relays. These services resolve hostnames for routing, network management, and security. The `Host` and `SNI` fields help route traffic to the correct server or service, ensuring load balancing and scalability in hybrid and multi-tenant environments.

Web Servers. CDN and LB services act as intermediaries between clients and backend servers, effectively masking clients from servers. Headers such as `X-Forwarded-For`, `X-Real-IP`, `X-Client-IP`, `True-Client-IP`, `Client-IP`, and `CF-Connecting-IP` transmit client information to backend servers [44], serving critical purposes such as logging and analytics. While these headers are intended to contain IPs, some web servers do not enforce this and resolve hostnames to IPs. Attackers can abuse them as DNS relays via header resolution. Other headers, such as `Referer` and `X-WAP-Profile`, which may carry URLs or IP addresses, are also susceptible to abuse.

Email Systems. Email servers resolve hostnames in the `HELO` or `FROM` email addresses to verify the authenticity and legitimacy of the sender's domain. As a result, it is common for most mail servers to perform DNS resolution. However, our evaluation indicates that not all mail servers can be abused for DNS relaying, as some incorporate verification processes to validate the authenticity of the sender's mail server and the existence of recipient email addresses. Additionally, web services that offer email sign-up or account registration may be exploited as DNS relays via email systems, as they rely on email verification to confirm ownership.

Content Fetching Applications. Applications that fetch content from user-provided URLs, such as Internet Archive, Slack, and X, involve DNS resolution, creating potential attack surfaces for DNS relaying. These platforms resolve URLs to enable features like snapshots, monitoring, and previews. However, this can be exploited by attackers by embedding tunneling domains in URLs. Slack, widely used for organizational communication, is particularly at risk.

Security Services. Security services often perform DNS resolution on domains as part of their security checks to gather additional information. For instance, VirusTotal may resolve domains to IP addresses and retrieve hosted content to detect malicious activities. Similarly, firewall solutions [11] may resolve hostnames to IP addresses to defend against domain

TABLE IV: Comparison of attack capabilities and surfaces between open resolvers and relay resolvers.

	Open Resolvers	Relay Resolvers	Comparison
Request ($C \rightarrow R$†) Request ($R \rightarrow N$)	Arbitrary format Receive request	Query name Receive request	Limited Equal
Response ($N \rightarrow R$) Response ($R \rightarrow C$)	Arbitrary format Receive response	Arbitrary format Response latency⋆	Equal Limited
Protocol	DNS	HTTP, HTTPS, HTTP/3, TLS, SMTP, QUIC	Enhanced
Prevalence	1.80M IPs	15.19M IPs	Enhanced
Reachability	Resolver	Service, Resolver	Enhanced

† C: Client, R: Resolver, N: Nameserver
⋆ Response latency is a unique and reliable covert channel discovered by us.

fronting and its variants [11], [14], [19], [71], which exploit CDN services to conceal malicious content.

VI. Enhanced DNS Attacks

In this section, we examine how DNS relaying enhances three specific DNS attack types: DNS tunneling, DNS amplification, and DNS exploit.

A. Relay Resolvers

DNS relaying introduces new attack opportunities by transforming legitimate services into *resolvers with limited capabilities*, referred to as relay resolvers, as detailed in Table IV. The DNS request-response process consists of four stages: client-to-resolver ($C \rightarrow R$), resolver-to-nameserver ($R \rightarrow N$), nameserver-to-resolver ($N \rightarrow R$), and resolver-to-client ($R \rightarrow C$). Relay resolvers are limited in the $C \rightarrow R$ stage, where attackers can only manipulate query names, and in the $R \rightarrow C$ stage, as they are unable to receive DNS responses. Despite these limitations, we uncover a reliable covert channel, namely *response latency*, that attackers can exploit to transmit data during the $R \rightarrow C$ stage.

Response latency covert channel. During our manual analysis, we observed that some affected services, particularly CDN providers like Akamai and Incapsula, exhibit response latencies closely tied to DNS resolution delays. These services often perform synchronous DNS lookups to fetch content and are optimized for low and predictable response times to ensure high quality of service for their tenants' end users. This optimization stabilizes the timing channel, making it a reliable vector for covert communication. In contrast, services relying on asynchronous lookups, common in content-fetching or security systems, tend to mask DNS resolution delays, making the covert channel harder to exploit. Similarly, services with naturally high processing latencies, such as email systems performing sender verification, can obscure resolution timing and reduce channel reliability.

To assess the prevalence of the response latency covert channel, we conduct experiments to measure DNS relay behavior under nameserver delays. For each relay, we issue two DNS queries for `attacker.tld`, configuring our nameserver to respond to one query while ignoring the other. The response times, denoted as T_a (nameserver answers) and T_{na} (nameserver does not answer), are recorded. Each experiment is repeated three times with different queries. A covert channel is considered present if the empirical condition $|T_a - T_{na}| \geq \min(T_a, T_{na})$ is satisfied, which requires the larger response time to be more than twice the smaller. Table V summarizes our findings across protocols: 89.3% of HTTP, 91.3% of HTTPS and 57.2% of TLS relays in top domain measurement exhibited an observable difference between T_a and T_{na}, indicating their potential for abuse in data infiltration. Interestingly, a few DNS relays, like `www.usm.edu`, had $T_a > T_{na}$ due to long processing times for normal requests and DNS resolution shortcuts that reduced T_{na}.

TABLE V: The ratio of DNS relays with latency covert channels from top domain measurement.

Protocol	# of Popular Domains	Latency Channel Ratio	
		$T_a < T_{na}$	$T_a > T_{na}$
HTTP	80,001	89.3%	1.7e-4
HTTPS	51,957	91.3%	9.5e-5
TLS	22,110	57.2%	4.5e-4

Enhancement. Although relay resolvers are limited in capability, they significantly enhance DNS attacks from protocol, prevalence, and reachability perspectives, as outlined in Table IV. From a protocol standpoint, relay resolvers can embed DNS tunneling payloads into non-DNS protocols such as HTTP, TLS, and SMTP, bypassing traditional DNS tunneling detection mechanisms that focus solely on DNS packets. From a prevalence perspective, the number of relay resolvers is 8 times greater than normal open resolvers, substantially expanding the attack surface for DNS amplification attacks and exploits targeting vulnerable DNS components. From a reachability perspective, relay resolvers enable attackers to craft malicious query names and DNS responses and extend their reach to additional vulnerable services, such as email servers, beyond just vulnerable resolvers.

B. Relayed DNS Tunneling

How DNS relaying revives DNS tunneling? Figure 6(a) presents the basic idea of legacy DNS tunneling. Although this technique has posed significant risks, defensive mechanisms deployed at enterprise resolvers and network boundaries have greatly reduced its impact [18], [34], [43], [47], [48], [70], [79]. However, malicious actors can leverage DNS relaying to achieve relayed DNS tunneling as shown in Figure 6(b) which uses Web services as an example. In this example, the exfiltrated data is put into the `Host` field of an HTTP header. When the Web service receives the HTTP header, it performs a DNS lookup for the domain in the `Host` field and the later steps follow legacy DNS exfiltration. In addition to the exfiltration channels over HTTP headers discovered in [10], we uncover more exfiltration channels, such as via TLS and SMTP. Moreover, although relayed data exfiltration is relatively straightforward, achieving data infiltration in DNS relaying is considerably more challenging. In this paper, we propose to use the response latency covert channel for data infiltration. In summary, using the Web service as a relay, malicious actors can evade the protective DNS security services deployed by enterprises. Unlike direct communica-

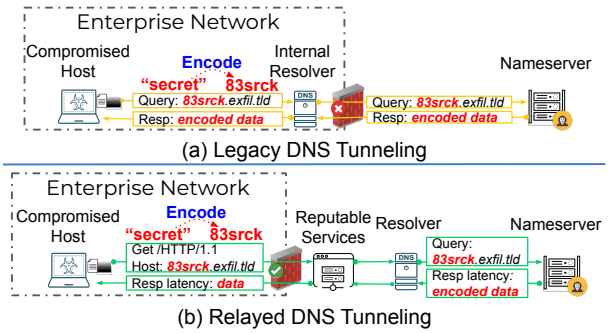


Fig. 6: Workflow of relayed DNS tunneling attacks.

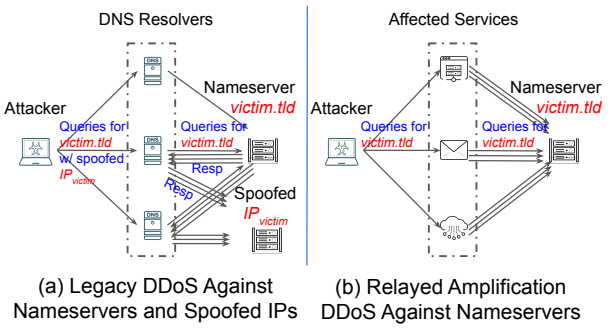


Fig. 7: Legacy DNS DDoS versus relayed amplification DDoS.

tion to malicious endpoints, which can be blocked based on destination, relayed DNS tunneling targets legitimate services and requires per-session content inspection for defense.

Relayed data exfiltration. To evaluate the data exfiltration capacity of DNS relays, we tested Akamai (HTTP `Host`) and Incapsula (HTTP `Host` and TLS `SNI`) by encoding data as `$data.attacker.tld` and increasing its length until our nameserver can no longer receive the query. RFC 1035 [40] defines the maximum length of hostnames as 255 octets, with a maximum of 63 octets per label. Both Akamai and Incapsula adhere to this RFC: their Host header is case-insensitive and can reach a maximum length of 253 characters (excluding one length prefix byte and one trailing null byte [63]); the SNI for Incapsula is case-sensitive and has a maximum length of 250 characters due to the probing of `www.$data.attacker.tld`. Let `$domain` denote the root domain which does not encode information. Thus, we can use Base32 encoding (5 bits per character) to exfiltrate 1,200 bits per request over the case-insensitive Host header, namely, $(253-length(\$domain)-4)\times 5 \approx 240\times 5 \approx 1,200$. We also employ Base64 encoding (6 bits per character) to exfiltrate 1,400 bits per request over the case-sensitive SNI field (ASCII characters such as +/= can be relayed), namely, $(250-length(\$domain)-4)\times 6 \approx 236\times 6 \approx 1,400$. Akamai and Incapsula enforce a limit of 60 requests per minute, but attackers could scale horizontally to increase throughput.

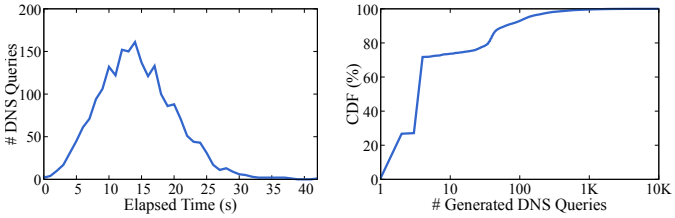
Relayed data infiltration. In §VI-A, we have discovered a novel and generic side channel, *response latency*, that can be exploited for data infiltration. For instance, when a nameserver is configured to respond immediately, Akamai and Incapsula exhibit response times of 0.2s and 0.5s, respectively. However, a non-responsive nameserver significantly increases these latencies to 4.6s and 36.6s, respectively. These variations in response times can be leveraged to encode information and achieve data infiltration. Let T_{min} and T_{max} denote the minimum and maximum response times of legitimate services. One intuitive encoding scheme is to configure the nameserver to delay its response by $N \times T_{min}$ seconds, where $N \in Z$ and $N \in [0, T_{max}/T_{min} - 1)$. This allows for T_{max}/T_{min} possible values of N, enabling the encoding of $\log_2 T_{max}/T_{min}$ bits per request. For Akamai and Incapsula, this scheme can theoretically encode 4 and 6 bits per request, respectively. Attackers can achieve a higher infiltration rate through horizontal scaling. Note that these estimations are conducted based on theoretical models and may vary in practice due to network conditions and payload sizes. We provide empirical measurements in Appendix A.

Proof-of-concept. Based on the insight that infiltrated commands are typically much shorter than exfiltrated data, a dynamic consistent with the latency channel's low infiltration rate and the query names' high exfiltration rate, we developed a tool to exploit Akamai and Incapsula servers for relayed DNS tunneling. A demonstration of this tool is available in a video.[2] The tool includes a mechanism for a compromised client to measure server latency and transmit the N value to an attacker controlled nameserver, enabling it to adapt to new DNS relays, changing network conditions, and varying payloads. In the video, the tool allows the attacker-controlled server to issue on-demand commands and navigate compromised systems in real time. We will release the tool's source code as part of our commitment to open science.[3]

C. Relayed DNS Amplification

How DNS relaying enhances DNS DDoS? Legacy DNS amplification attacks, shown in Figure 7(a), are increasingly difficult to execute due to stricter IP spoofing defenses [23], [61], and the shrinking number of open resolvers [30], [32], [49], [77]. Widely adopted defenses like ACLs [45], RRL [3], and prompt patching further limit their effectiveness and adoption [76]. DNS relaying enhances DNS DDoS with a new type of amplifiers as shown in Figure 7(b). Initially, it appears to only provide relay resolvers, with no ability for attackers to specify query types, receive responses, or spoof source IPs. However, DNS relays can trigger aggressive lookups when authoritative servers do not respond, turning them into promising amplifiers. For example, in Figure 8(a), a single HTTPS request can generate up to 2,139 DNS queries in 42 seconds. In comparison, the attacker's cost involves 21 packets, which include TCP and TLS handshakes, HTTP request and response, and connection teardown, with the HTTP request and response requiring only 4 packets. Attackers can further exploit DNS relay amplifiers by maintaining connections and issuing distinct queries to spread out the costs associated with TCP and TLS handshakes. This strategy allows them

[2]Link to the Akamai demo video: https://youtu.be/5CenBUGHCRQ.
[3]Link to the tool: https://github.com/paloaltonetworks/dns-relay-c2



(a) A single HTTPS request can trigger 2,139 DNS queries.

(b) The CDF of DNS query amount generated per request.

Fig. 8: #DNS queries in top domain measurement.

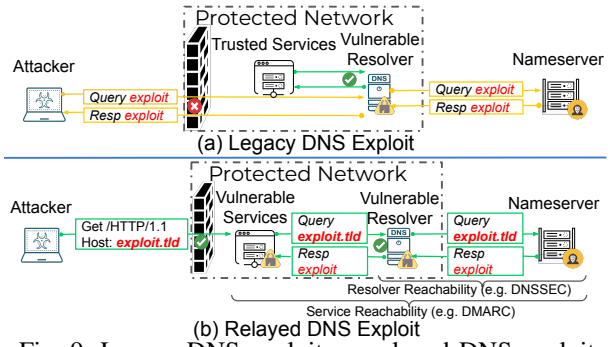


Fig. 9: Legacy DNS exploit vs. relayed DNS exploit.

to achieve a packet amplification factor of 535 and a byte amplification factor of 599. Additionally, Figure 8(b) shows that 10% of relays in top domain measurement can generate over 62 DNS queries, with an average of 122 queries per request. In the Internet-wide measurement, results show that DNS relays incur an average of 14 DNS packets per probe, with HTTPS probes on port 22 generating the most, averaging 149 packets. We report the number of DNS queries rather than standard packet or byte amplification factors, as these metrics depend on relay-specific traffic sizes and rate limits.

Relayed DNS amplification. We describe two approaches to launch DNS amplification attacks by employing request amplifiers and nameserver configurations. The first, *DNS-specific amplification*, leverages DNS relays to amplify traffic between resolvers and victim nameservers. Attackers issue queries for hostnames under `victim.tld`, generating amplified requests toward the victim's nameservers. For resolvers susceptible to NXNSAttack [1], queries under `attacker.tld` can be replied with numerous non-existent NS responses pointing to `victim.tld`, further boosting traffic amplification. The second, *IP-focused amplification*, targets arbitrary victim IPs via NS records. By creating entries such as `ns.attacker.tld A` IP_{victim} at DNS hosting providers, relays can be abused to flood chosen IPs. Leveraging DNS relays as amplifiers offers several advantages. First, their high uplink bandwidth indicates a single DNS relay can generate a huge number of DDoS traffic, theoretically. Moreover, DNS relays can have high amplification factors. Finally, there are a huge amount of DNS relays across the Internet.

D. Relayed DNS Exploit

Many vulnerable DNS components are unreachable externally. A critical precondition for legacy DNS exploit attacks, shown in Figure 9(a), is that *vulnerable DNS servers are reachable from public networks*. Unfortunately, in many cases, these servers are located in isolated networks that are unreachable externally. Table IV shows that DNS relaying expands the attack surface for DNS exploit-based attacks in two ways. First, relay resolvers increase the number of reachable server IPs, being 8 times more common than open resolvers. Second, they allow external attackers to interact with internal DNS servers (*resolver reachability*) and internal services dependent on them (*service reachability*), also known as DNS clients. However, as shown in Figure 9(b), attackers face limitations when exploiting relay resolvers: they can only specify query names of DNS requests and cannot directly receive DNS responses. From request perspective, this limitation still allows exploits using malformed query names with special characters (e.g. +/=). From the response perspective, although attackers cannot directly access DNS responses, these responses can still enable both *resolver reachability* and *service reachability*. For example, an attacker could trigger a DNS query from internal servers to a malicious domain with an attacker-controlled authoritative nameserver, crafting arbitrary malformed responses to target internal DNS servers, thereby achieving *resolver reachability*. Attackers can leverage *resolver reachability* to launch brute-force attacks [37], [38] or exploit vulnerabilities in resolver-specific functionalities that are independent of query type. For instance, resolvers will follow delegation (e.g., NS) [1], aliasing (e.g., CNAME, DNAME) [29], [77], and perform integrity verification (e.g., SIG, DNSKEY, RRSIG) [17], [67], consuming the corresponding response types. However, mismatches between query and response types often prevent these responses from reaching upstream services. If specific query types (e.g., CNAME, TXT, MX, SPF) can be triggered, as shown with Incapsula and mail servers in Figure 3(c), attackers can craft matching response types to exploit services that require *service reachability* [2], [22], [26], [27], [57], [69], [78]. Below is a CNAME record designed for a cache poisoning attack against `victim.tld`, where the zero byte (\000) signifies the end of the data [26]. We do not issue crafted responses to DNS relays to test their vulnerabilities, to stay ethical and avoid disrupting legitimate services.

```
zerobyte.attacker.tld CNAME victim.tld\000.attacker.tld
victim.tld\000.attacker.tld A 1.2.3.4
```

VII. Discussion

A. Coverage Analysis

Our attack surface measurements and root cause analysis represent best-effort estimates and should be interpreted as a *lower bound*, as some attack surfaces or use cases may remain undiscovered. Results can vary with testing time, source IPs, and experiment setup. For example, HTTP/HTTPS/HTTP/3 scans may miss affected services if DNS resolution depends on specific paths, headers, or field combinations. SMTP coverage was limited by blocks (e.g., Spamhaus [55]) and recipient validation, likely underestimating affected mail servers. Other protocols or ports, such as LDAP using SRV records [26], as

well as internal network deployments, may also be vulnerable. Some observations may stem from middleboxes (e.g., firewalls or ISP-managed devices) [39], [58], [74]. When their behavior is deterministic, such middleboxes effectively act as DNS relays. We argue that their presence does not affect our core conclusions, and leave middlebox identification as future work.

B. Responsibilities and Defenses

DNS relaying is not inherently a vulnerability, but can enable legacy DNS attacks to bypass modern security perimeters. Its role in attacks raises important questions about stakeholder responsibility. We outline actions that stakeholders can take to protect themselves from potential attackers.

Input Sanitization and Access Control. Most DNS relays arise from legitimate services. The primary defense is to reduce or eliminate unnecessary DNS resolutions. Stakeholders, including CDN providers, cloud platforms, and web servers, should sanitize client input, apply source IP filtering or authenticated access, limit resolver retries, and maintain updated DNS clients, forwarders, and resolvers.

Protective DNS. When DNS lookups are unavoidable (e.g., email systems, content fetching applications, security services), maintainers should deploy protective DNS solutions [18], [35], [59], [70], [79]. Services that accept user-controlled input should also ensure their backend resolvers are promptly patched and properly secured.

Full Traffic Inspection. Because legitimate services may not act quickly to reduce DNS relays, enterprise and ISP stakeholders can adopt full traffic inspection to detect and block attacks that leverage relayed DNS activity, particularly DNS tunneling, in enterprise networks.

C. Ethical Considerations

For network scanning, we deployed the same techniques used in prior studies [12], [13], [24], [52]. To reduce our scanning impact, we send only one request to each destination and limit the scanning rate. We also published a website and TXT records for the scan domain to outline our objectives and provide contacts. We received several inquiries and added their relevant domains/IPs to our scan opt-out list.

We reported DNS relaying risks to 15 major vendors. Among them, Akamai and Incapsula account for most HTTP/HTTPS/TLS-based DNS relays in our top domain measurements. We conducted two disclosure rounds: the data exfiltration risk to Akamai and Incapsula in May 2024 and bi-directional C2 risk to all vendors in March 2025. In the first round, Akamai responded that data exfiltration is not a real security concern. In the second round, after we demonstrated efficient and reliable C2, they acknowledged the security issue and collected our code/data for internal evaluation and mitigation planning. Google and Amazon likewise acknowledged our findings. Incapsula treated this issue as abuse. We identified this as a major security threat due to C2 potential, scale, and in-the-wild abuse, and implemented full traffic inspection.

VIII. Conclusion

We present DNS relaying, a foundational technique that broadens the scope of DNS-based attacks by exploiting legitimate services as *limited-capability resolvers*. Through a novel response latency covert channel, DNS relaying enables bi-directional DNS tunneling, with 1,400 bits exfiltration and 6 bits infiltration per request. Besides, we show that DNS relays can achieve a packet amplification factor of up to 535, and expose otherwise unreachable DNS components to attackers, increasing exploitation risks. Our Internet-wide scan reveals 15.19M DNS relay IPs, 8 times more than open resolvers. We responsibly disclosed our findings; Akamai, Google, and Amazon confirmed the risks and initiated evaluations.

Appendix

A. Infiltration Rate Measurements

We measure the empirical infiltration rate of the latency covert channel across temporal windows and source geolocations. We deploy client VMs in five geographic regions (US-Central, US-East, US-West, EU, Asia-Pacific) and probe two Akamai and two Incapsula HTTP relay IPs in the US per region over one day, collecting three response-latency samples per delay level every 4 hours. Figures 10(a)–(b) show RTT distributions per server delay level for Akamai (US-Central) and Incapsula (US-West), confirming linear growth; ∞ marks the relay HTTP timeout ($T_{\max}$ of 2.1–3.1 s for Akamai, 34–42 s for Incapsula). At demo time, Akamai used a fixed $\approx$1.2 s DNS retry timeout, yielding $T_{\max}\approx$4.4 s. At measurement time, the retry appears randomized in $[0.1, 1.2]$ s, which may reflect a mitigation following our responsible disclosure; this shrinks the usable window and the number of distinguishable levels, suggesting a mitigation strategy applicable to all relay vendors. Figure 10(c) shows rates by time-of-day: Akamai varies from 2.0 bps (00–04 UTC) to 3.2 bps (04–08 UTC); Incapsula remains stable at 4.3–4.5 bps. Figure 10(d) shows rates by source geolocation: Akamai delivers 3.0–3.3 bps and Incapsula 4.3–4.6 bps, with Incapsula consistently higher due to its wider relay timeout window. The worst-case geolocation sustains 3.0 bps.

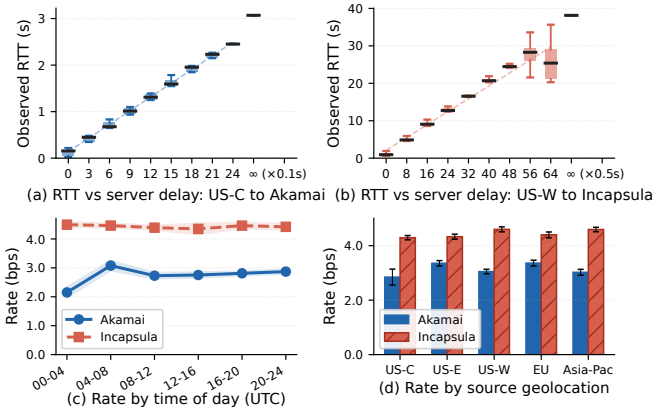


Fig. 10: Infiltration rate measurements. (a) Akamai, (b) Incapsula RTT vs. server delay; (c) by time of day; (d) by source geolocation.

REFERENCES

[1] Yehuda Afek, Anat Bremler-Barr, and Lior Shafir. {NXNSAttack}: Recursive {DNS} inefficiencies and vulnerabilities. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 631–648, 2020.

[2] Md. Ishtiaq Ashiq, Weitong Li, Tobias Fiebig, and Taejoong Chung. SPF beyond the standard: Management and operational challenges in practice and practical recommendations. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3081–3098, Philadelphia, PA, August 2024. USENIX Association. https://www.usenix.org/conference/usenixsecurity24/presentation/ashiq.

[3] CISA. Dns amplification attacks, 2024. https://www.cisa.gov/news-events/alerts/2013/03/29/dns-amplification-attacks.

[4] Cisco. Umbrella popularity list, 2024. https://umbrella-static.s3-us-west-1.amazonaws.com/index.html.

[5] Alibaba Cloud. Configure the default origin host - alibaba cloud documentation, 2025. https://www.alibabacloud.com/help/en/cdn/user-guide/configure-the-default-origin-host.

[6] Alibaba Cloud. Server load balancer - alibaba cloud, 2025. https://www.alibabacloud.com/en/product/server-load-balancer.

[7] Google Cloud. Application load balancer overview, 2025. https://cloud.google.com/load-balancing/docs/application-load-balancer.

[8] Tencent Cloud. Combining sni proxy with dns resolution, 2025. https://cloud.tencent.com/developer/article/2322792.

[9] Tencent Cloud. Sni support for binding multiple certificates to a clb instance, 2025. https://www.tencentcloud.com/document/product/214/19048.

[10] The Contractor. Data-bouncing - the art of indirect exfiltration. using & abusing trusted domains as a 2nd order transport., 9 2023. https://thecontractor.io/data-bouncing/.

[11] Tianze Ding and Junyu Zhou. Domain borrowing: Catch my c2 traffic if you can, 2021. https://www.blackhat.com/asia-21/briefings/schedule/#domain-borrowing-catch-my-c2-traffic-if-you-can-22314.

[12] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J Alex Halderman. A search engine backed by internet-wide scanning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 542–553, 2015.

[13] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. {ZMap}: Fast internet-wide scanning and its security applications. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 605–620, 2013.

[14] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies*, 2015.

[15] FireEye. Highly evasive attacker leverages solarwinds supply chain to compromise multiple global victims with sunburst backdoor. *FireEye*, 2020. https://cloud.google.com/blog/topics/threat-intelligence/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.

[16] Fortinet. Multiple threats target adobe coldfusion vulnerabilities, 2025. https://www.fortinet.com/blog/threat-research/multiple-threats-target-adobe-coldfusion-vulnerabilities.

[17] Elias Heftrig, Haya Schulmann, Niklas Vogel, and Michael Waidner. The harder you try, the harder you fail: The keytrap denial-of-service algorithmic complexity attacks on dnssec. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 497–510, 2024.

[18] Artsiom Holub. Introducing improvements in dns tunneling & dns exfiltration detection, 2023. https://umbrella.cisco.com/blog/improvements-dns-tunneling-dns-exfiltration-detection.

[19] Erik Hunstad. Domain fronting is dead, long live domain fronting, 2020. https://github.com/SixGenInc/Noctilucent.

[20] Imperva. Automatic domain validation for imperva-generated certificates, 2025. https://docs.imperva.com/bundle/cloud-application-security/page/cname-account.htm.

[21] Infoblox. Decoy dog is no ordinary pupy: Separating a sly dns malware from the pack, 2023. https://blogs.infoblox.com/threat-intelligence/decoy-dog-is-no-ordinary-pupy-distinguishing-malware-via-dns/.

[22] Zero Day Initiative. Exim libspf2 integer underflow remote code execution vulnerability, 2023. https://www.zerodayinitiative.com/advisories/ZDI-23-1472/.

[23] Internet Society. Mutually agreed norms for routing security (manrs), 2024. https://www.manrs.org/.

[24] Liz Izhikevich, Renata Teixeira, and Zakir Durumeric. {LZR}: Identifying unexpected internet services. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3111–3128, 2021.

[25] Bahruz Jabiyev, Omid Mirzaei, Amin Kharraz, and Engin Kirda. Preventing server-side request forgery attacks. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 1626–1635, 2021.

[26] Philipp Jeitner and Haya Shulman. Injection attacks reloaded: Tunnelling malicious payloads over DNS. In *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[27] Philipp Jeitner, Haya Shulman, Lucas Teichmann, and Michael Waidner. {XDRI} attacks-and-how to enhance resilience of residential routers. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4473–4490, 2022.

[28] jozo. Hundreds of crew members fired for using free internet access app onboard, 2018. https://crew-center.com/hundreds-crew-members-fired-using-free-internet-access-app-onboard.

[29] Siva Kesava Reddy Kakarla, Ryan Beckett, Todd Millstein, and George Varghese. {SCALE}: Automatically finding {RFC} compliance bugs in {DNS} nameservers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 307–323, 2022.

[30] Marc Kührer, Thomas Hupperich, Jonas Bushart, Christian Rossow, and Thorsten Holz. Going wild: Large-scale classification of open dns resolvers. In *Proceedings of the 2015 Internet Measurement Conference*, pages 355–368, 2015.

[31] 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 *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3153–3170, 2023.

[32] Xiang Li, Dashuai Wu, Haixin Duan, and Qi Li. Dnsbomb: A new practical-and-powerful pulsing dos attack exploiting dns queries-and-responses. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 253–253. IEEE Computer Society, 2024.

[33] Xiang Li, Wei Xu, Baojun Liu, Mingming Zhang, Zhou Li, Jia Zhang, Deliang Chang, Xiaofeng Zheng, Chuhan Wang, Jianjun Chen, et al. Tudoor attack: Systematically exploring and exploiting logic vulnerabilities in dns response pre-processing with malformed packets. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 181–181. IEEE Computer Society, 2024.

[34] Daiping Liu, Ruian Duan, and Jun Wang. Resolution without dissent: In-path per-query sanitization to defeat surreptitious communication over dns. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 4–4. IEEE Computer Society, 2024.

[35] Mingxuan Liu, Zhang Yiming, Xiang Li, Chaoyi Lu, Baojun Liu, Haixin Duan, and Xiaofeng Zheng. Understanding the implementation and security implications of protective dns services. In *Proc. Netw. Distrib. Syst. Secur. Symp*, 2024.

[36] PortSwigger Ltd. Burp collaborator, 2025. https://portswigger.net/burp/documentation/desktop/tools/collaborator.

[37] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. Dns cache poisoning attack reloaded: Revolutions with side channels. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1337–1350, 2020.

[38] Keyu Man, Xin'an Zhou, and Zhiyun Qian. Dns cache poisoning attack: Resurrections with side channels. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3400–3414, 2021.

[39] Ariana Mirian, Alisha Ukani, Ian Foster, Gautam Akiwate, Taner Halicioglu, Cynthia T Moore, Alex C Snoeren, Geoffrey M Voelker, and Stefan Savage. In the line of fire: Risks of dpi-triggered data collection. In *Proceedings of the 16th Cyber Security Experimentation and Test Workshop*, pages 57–63, 2023.

[40] Paul V Mockapetris. Rfc1035: Domain names-implementation and specification, 1987.

[41] Elizabeth Montalbano. Report: Air-gapped networks vulnerable to dns attacks, 2022. https://www.darkreading.com/attacks-breaches/report-air-gapped-networks-vulnerable-dns-attacks.

[42] Giovane CM Moura, Sebastian Castro, John Heidemann, and Wes Hardaker. Tsuname: exploiting misconfiguration and vulnerability to ddos dns. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 398–418, 2021.

[43] Asaf Nadler, Avi Aminov, and Asaf Shabtai. Detection of malicious and low throughput data exfiltration over the dns protocol. *Computers & Security*, 80:36–53, 2019.

[44] Mozilla Developer Network. The http x-forwarded-for (xff) request header is a de-facto standard header for identifying the originating ip address of a client connecting to a web server through a proxy server., 2025. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-For.

[45] Yevheniya Nosyk, Maciej Korczyński, Qasim Lone, Marcin Skwarek, Baptiste Jonglez, and Andrzej Duda. The closed resolver project: Measuring the deployment of inbound source address validation. *IEEE/ACM Transactions on Networking*, 31(6):2589–2603, 2023.

[46] OWASP Top 10 team. A10:2021 – server-side request forgery (ssrf). Online, 2025. https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_(SSRF)/.

[47] Yarin Ozery. Akamai develops real-time detections for dns exfiltration. https://www.akamai.com/blog/security/akamais-real-time-detections-for-dns-exfiltration.

[48] Yarin Ozery, Asaf Nadler, and Asaf Shabtai. Information based heavy hitters for real-time dns data exfiltration detection. In *Proc. Netw. Distrib. Syst. Secur. Symp*, pages 1–15, 2024.

[49] Jeman Park, Rhongho Jang, Manar Mohaisen, and David Mohaisen. A large-scale behavioral analysis of the open dns resolvers on the internet. *IEEE/ACM Transactions on Networking*, 30(1):76–89, 2021.

[50] Giancarlo Pellegrino, Onur Catakoglu, Davide Balzarotti, and Christian Rossow. Uses and abuses of server-side requests. In *Research in Attacks, Intrusions, and Defenses: 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings 19*, pages 393–414. Springer, 2016.

[51] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. *arXiv preprint arXiv:1806.01156*, 2018.

[52] Damian Poddebniak, Fabian Ising, Hanno Böck, and Sebastian Schinzel. Why {tls} is better without {starttls}: A security analysis of {starttls} in the email context. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 4365–4382, 2021.

[53] PortSwigger. Oast: Out-of-band application security testing, August 2017. https://portswigger.net/blog/oast-out-of-band-application-security-testing.

[54] PortSwigger. Finding and exploiting blind xxe vulnerabilities, 2025. https://portswigger.net/web-security/xxe/blind.

[55] Spamhaus Project. Spamhaus: Strengthening trust and safety across the internet, 2025. https://www.spamhaus.org/.

[56] ProjectDiscovery. An oob interaction gathering server and client library, 2025. https://github.com/projectdiscovery/interactsh.

[57] Pedro Ribeiro and Radek Domanski. (pwn2own) tp-link archer a7 dns response stack-based buffer overflow remote code execution vulnerability, 2020. https://www.zerodayinitiative.com/advisories/ZDI-20-333/.

[58] Laura M Roberts and David Plonka. Watching the watchers: Nonce-based inverse surveillance to remotely detect monitoring. *Network Traffic Measurement and Analysis Conference*, 2020.

[59] Elsa Rodríguez, Radu Anghel, Simon Parkin, Michel Van Eeten, and Carlos Gañán. Two sides of the shield: Understanding protective {DNS} adoption factors. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3135–3152, 2023.

[60] Sandip Saha, Sareena Karapoola, Chester Rebeiro, et al. Yoda: Covert communication channel over public dns resolvers. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 252–260. IEEE, 2023.

[61] Daniel Senie and Paul Ferguson. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827, May 2000. https://www.rfc-editor.org/info/rfc2827.

[62] Amazon Web Services. Listeners for your application load balancers, 2025. https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-listeners.html.

[63] StackExchange. What is the maximum length of a domain name?, 2024. https://superuser.com/questions/1843857/what-is-the-maximum-length-of-a-domain-name.

[64] Daniel Stenberg. Http/3 support in curl is considered experimental until further notice when built to use quiche or msh3. only the ngtcp2 backend is not experimental., 2024. https://curl.se/docs/http3.html.

[65] Akamai Technologies. An origin server is a physical location that houses your deliverable content like a site or app., 2025. https://techdocs.akamai.com/property-mgr/docs/origin-server.

[66] Aleksei Tiurin. Ssrf vulnerabilities caused by sni proxy misconfigurations, 2022. https://www.invicti.com/blog/web-security/ssrf-vulnerabilities-caused-by-sni-proxy-misconfigurations/.

[67] Sagi Tzadik. Sigred – resolving your way into domain admin: Exploiting a 17 year-old bug in windows dns servers, 2020. https://research.checkpoint.com/2020/resolving-your-way-into-domain-admin-exploiting-a-17-year-old-bug-in-windows-dns-servers/.

[68] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. Dnssec and its potential for ddos attacks: a comprehensive measurement study. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 449–460, 2014.

[69] VulDB. Exim dmarc dmarc.c dmarc_dns_lookup use after free, 2022. https://vuldb.com/?id.211919.

[70] Shu Wang, Daiping Liu, and Ruian Duan. Leveraging dns tunneling for tracking and scanning, 2024. https://unit42.paloaltonetworks.com/three-dns-tunneling-campaigns/.

[71] Mingkui Wei. Domain shadowing: Leveraging content delivery networks for robust Blocking-Resistant communications. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3327–3343. USENIX Association, 2021. https://www.usenix.org/conference/usenixsecurity21/presentation/wei.

[72] Malte Wessels, Simon Koch, Giancarlo Pellegrino, and Martin Johns. SSRF vs. developers: A study of SSRF-Defenses in PHP applications. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 6777–6794, Philadelphia, PA, August 2024. USENIX Association. https://www.usenix.org/conference/usenixsecurity24/presentation/wessels.

[73] Kyle Wilhoit and Robert Falcone. Oilrig uses updated bondupdater to target middle eastern government, 2018. https://unit42.paloaltonetworks.com/unit42-oilrig-uses-updated-bondupdater-target-middle-eastern-government/.

[74] Yunpeng Xing, Chaoyi Lu, Baojun Liu, Haixin Duan, Junzhe Sun, and Zhou Li. Yesterday once more: Global measurement of internet traffic shadowing behaviors. In *Proceedings of the 2024 ACM on Internet Measurement Conference*, pages 230–240, 2024.

[75] Wei Xu, Xiang Li, Chaoyi Lu, Baojun Liu, Haixin Duan, Jia Zhang, Jianjun Chen, and Tao Wan. Tsuking: Coordinating dns resolvers and queries into potent dos amplifiers. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 311–325, 2023.

[76] Omer Yoachimik and Jorge Pacheco. Targeted by 20.5 million ddos attacks, up 358% year-over-year: Cloudflare's 2025 q1 ddos threat report, April 2025. https://blog.cloudflare.com/ddos-threat-report-for-2025-q1/.

[77] Qifan Zhang, Xuesong Bai, Xiang Li, Haixin Duan, Qi Li, and Zhou Li. {ResolverFuzz}: Automated discovery of {DNS} resolver vulnerabilities with {Query-Response} fuzzing. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4729–4746, 2024.

[78] Xiaofeng Zheng, Chaoyi Lu, Jian Peng, Qiushi Yang, Dongjie Zhou, Baojun Liu, Keyu Man, Shuang Hao, Haixin Duan, and Zhiyun Qian. Poison over troubled forwarders: A cache poisoning attack targeting {DNS} forwarding devices. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 577–593, 2020.

[79] Michael Zuckerman. Dns early detection - cobalt strike dns c2, 2024. https://blogs.infoblox.com/threat-intelligence/dns-early-detection-cobalt-strike-dns-c2/.