Summer 2017

On The Impact of Internet Naming Evolution: Deployment, Performance, and Security Implications

Shuai Hao
College of William and Mary - Arts & Sciences, haoscs@gmail.com

On the Impact of Internet Naming Evolution: Deployment, Performance, and Security Implications

Shuai Hao

Tianjin, China

Master of Science, Beijing University of Posts and Telecommunications, 2010
Master of Science, London South Bank University, 2008
Bachelor of Engineering, North China Electric Power University, 2007

A Dissertation presented to the Graduate Faculty
of The College of William & Mary in Candidacy for the Degree of
Doctor of Philosophy

Department of Computer Science

College of William & Mary
January 2018

APPROVAL PAGE

This Dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

Shuai Hao

Approved by the Committee, January 2018

Committee Chair
Adjunct Professor Haining Wang, Computer Science
College of William & Mary

Professor Evgenia Smirni, Computer Science
College of William & Mary

Adjunct Associate Professor Kun Sun, Computer Science
College of William & Mary

Associate Professor Gang Zhou, Computer Science
College of William & Mary

Professor Chase Cotton, Electrical and Computer Engineering
University of Delaware

ABSTRACT

As one of the most critical components of the Internet, the Domain Name System (DNS) provides naming services for Internet users, who rely on DNS to perform the translation between the domain names and network entities before establishing an Internet connection. In this dissertation, we present our studies on different aspects of the naming infrastructure in today's Internet, including DNS itself and the network services based on the naming infrastructure such as Content Delivery Networks (CDNs).

We first characterize the evolution and features of the DNS resolution in web services under the emergence of third-party hosting services and cloud platforms. At the bottom level of the DNS hierarchy, the authoritative DNS servers (ADNSes) maintain the actual mapping records and answer the DNS queries. The increasing use of upstream ADNS services (i.e., third-party ADNS-hosting services) and Infrastructure-as-a-Service (IaaS) clouds facilitates the deployment of web services, and has been fostering the evolution of the deployment of ADNS servers. To shed light on this trend, we conduct a large-scale measurement to investigate the ADNS deployment patterns of modern web services and examine the characteristics of different deployment styles, such as performance, life-cycle of servers, and availability. Furthermore, we specifically focus on the DNS deployment for subdomains hosted in IaaS clouds.

Then, we examine a pervasive misuse of DNS names and explore a straightforward solution to mitigate the performance penalty in DNS cache. DNS cache plays a critical role in domain name resolution, providing (1) high scalability at Root and Top-level-domain nameservers with reduced workloads and (2) low response latency to clients when the resource records of the queried domains are cached. However, the pervasive misuses of domain names, e.g., the domain names of "one-time-use" pattern, have negative impact on the effectiveness of DNS caching as the cache has been filled with those entries that are highly unlikely to be retrieved. By leveraging the domain name based features that are explicitly available from a domain name itself, we propose simple policies for improving DNS cache performance and validate their efficacy using real traces.

Finally, we investigate the security implications of a fundamental vulnerability in DNS-based CDNs. The success of CDNs relies on the mapping system that leverages the dynamically generated DNS records to distribute a client's request to a proximal server for achieving optimal content delivery. However, the mapping system is vulnerable to malicious hijacks, as it is very difficult to provide pre-computed DNSSEC signatures for dynamically generated records in CDNs. We illustrate that an adversary can deliberately tamper with the resolvers to hijack CDN's redirection by injecting crafted but legitimate mappings between end-users and edge servers, while remaining undetectable by existing security practices, which can cause serious threats that nullify the benefits offered by CDNs, such as proximal access, load balancing, and DoS protection. We further demonstrate that DNSSEC is ineffective to address this problem, even with the newly adopted ECDSA that is capable of achieving live signing for dynamically generated DNS records. We then discuss countermeasures against this redirection hijacking.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I would first like to sincerely thank my advisor, Professor Haining Wang, for his guidance and support throughout my Ph.D. study. I feel very fortunate to work with Dr. Wang and his interests and enthusiasm in pursuing wonderful research continue to impact and inspire me for my future career.

I would like to thank my dissertation committee members, Prof. Evgenia Smirni, Prof. Gang Zhou, Prof. Kun Sun, and Prof. Chase Cotton, for their collective time and valuable insight.

All members of our research group team, Xing Gao, Dachuan Liu, Daiping Liu, Jidong Xiao, Haitao Xu, Zhang Xu, and Yubao Zhang, are making my Ph.D. journey a wonderful academic experience for exploring various ideas as well as an enjoyable, varied life for a lot of fun .

I would also like to thank Mr. Qunyang Lin and Dr. Junqing Xie at HP Labs China, and Dr. Junghwan Rhee and Dr. Chung Hwan Kim at NEC Laboratories America for offering me the research internships during my Ph.D. study.

Finally, I greatly thank my parents for their concern and support throughout my life. Also, I thank my sister for always believing in me. It's the best thing in my life relishing the moments of your growing up.

LIST OF TABLES

LIST OF FIGURES

On the Impact of Internet Naming Evolution: Deployment, Performance, and Security Implications

Chapter 1

Introduction

Accessing Internet services relies on the naming infrastructure, i.e., the Domain Name System (DNS), for translating human-readable domain names to routable network addresses. DNS is designed as a globally distributed database system, where the Root and Top-Level-Domain (TLD) nameservers are mainly used as the querying referrals, and the authoritative DNS servers (ADNSes) are responsible for storing the name-to-address records and returning answers to clients. Meanwhile, the acquired mapping results are cached by local DNS servers (LDNSes) to answer the following queries in a specific duration. The first two studies presented in this dissertation involve with DNS itself. In the first work, we present a large-scale measurement study to understand the DNS deployment for web services. Then, in the second work, we explore the simple policies to improve the effectiveness of DNS caching by considering the prevalent non-malicious misuse of domain names on the Internet.

Many Internet services have relied on DNS to achieve their enhanced functionalities. The most common example is the Content Delivery Networks (CDNs). CDNs play an important role in the Internet ecosystem by delivering a large fraction of the Internet content to end users with high availability, performance, and scalability. Typically, CDNs place a large number of edge servers at geographically distributed edge networks, enabling content caching and proximal access for end-users. The user requests for the content

hosted by CDNs are served at the "edge" via request redirection to improve user-perceived performance and balance the load across server clusters. Moreover, CDNs are able to provide a security portal of protection mechanism against distributed denial-of-service (DDoS) attacks by redirecting users from overwhelmed nodes [29, 123]. The third study in this dissertation aims to investigate a fundamental vulnerability that could be exploited by adversaries in a very stealthy manner to manipulate the access of end-users for CDNs and nullify the benefits offered by CDN vendors. We characterize popular CDN venders on the vulnerability and discuss countermeasures against such a security threat.

In sub-sections 1.1 - 1.3, we briefly introduce the motivation and problem statement of each study. In sub-section 1.4, we describe the organization of this dissertation.

1.1 Measuring and Characterizing the DNS deployment of Modern Web Services

Deploying authoritative nameservers requires extra hardware resources and additional maintenance support. Also, the critical roles of DNS services in web infrastructures make DNS an attractive target to attackers. Thus, web service providers are increasingly adopting the upstream authoritative DNS servers, including the top sites (e.g., Amazon and Twitter) that have the ability to maintain their own ADNS infrastructures. In addition, to save a large amount of investments for computing infrastructures, many popular web services are directly built upon Infrastructure-as-a-Service (IaaS) clouds such as Amazon EC2 and Windows Azure. The traditional web service providers are also migrating extended services into clouds to use the "illusively infinite" computing and storage resources. The IaaS infrastructure greatly facilitates the establishment of modern web services and also promotes the process of delegating the authoritative name resolution to third-party ADNS service providers. Besides traditional web-hosting providers such as Dyn [16] and Ultradns [35], the CDN and cloud service providers also offer the ADNS services that integrate the name resolution into their CDNs or cloud infrastructures [1, 5].

The existing DNS measurements focus on the characteristics of DNS activities and operations [50, 34, 61, 26, 65] at the root or top-level-domain servers [48, 51, 88, 90, 126], as well as the DNS resolvers [19, 36, 107]. Some works involving with the characteristics of ADNSes mainly center on the comparison with local DNS (LDNS) servers, but none of them explore various ADNS deployments for web services. Complementary to these prior works, we present a large-scale measurement study in attempt to answer the following questions: (1) how do modern web services deploy their ADNS servers? (2) what are the characteristics of different ADNS deployment patterns? and (3) in particular, how do the cloud-hosting subdomains administer their ADNS servers?

1.2 Exploring Domain Name Based Features on the Effectiveness of DNS Caching

Since DNS is a globally distributed database system, caching has been widely adopted in DNS infrastructures, where the acquired mapping results (i.e., DNS resource records, RRs) will be cached locally to answer the following queries in a specific duration. DNS cache significantly reduces the resolution traffic along referral chains to interact with multiple name servers, resulting in much shortened client-perceived delay and high scalability of DNS.

Due to its fundamental role for accessing Internet services, DNS traffic is the least blocked [103], and provides both attackers and developers with an attractive channel to transmit information. Thus, the misuse of domain names (either malicious or non-malicious) is widely observed on the Internet. On the other hand, since the cached objects in DNS resolvers are typically small, in some instances the caches are not size limited [79] and the memory usage is relatively stable as the expired entries are being evicted. However, when serving a large group of users with heavy workload (e.g., in ISP or CDN/cloud providers), although modern DNS resolvers manage the memory well, the cache will still quickly consume the memory bytes and go into swap. Meanwhile, this may also cause

performance problems on CPU if *cleaning-interval* is enabled to check the stale records periodically. To this effect, a fixed memory allocation is a common configuration [54] and the typical replacement policies (e.g., LRU and LFU) are employed to manage the cache usage [56, 64]. Therefore, it is critical to ensure that the cached RRs would be quite likely to be accessed again. Unfortunately, the pervasiveness of misused domains, e.g., *disposable domains* [54], has caused the ineffectiveness of caching on resolvers since the cache is filled up by the records with very low or almost zero cache hit rates.

In this work, we attempt to mitigate the negative effect on DNS caching caused by the domain name misuses, especially the "one-time-use" domains. Different from previous approaches, we do not pursue an accurate detection of domain misuses by employing the deep inspection techniques such as behavioral features [47], alphanumeric characters-based metric [106], or entropy-based computing [103]. Instead, our key insight is that most misused domains, either malicious or benign, tend to transmit information over DNS query names. Thus, the domain name itself may have distinct features that are explicitly available from individual queries and can be readily exploited for improving DNS cache performance. By extracting the *re-used* and *once-used* RRs from our trace logs captured at the campus networks of two universities, we propose and validate the explicit domain name-based features that could be used to improve the effectiveness of DNS caches by proactively eliminating the domains that are highly unlikely to be reused in future's queries.

1.3 End Users Get Maneuvered: Redirection Hijacking in Content Delivery Networks

CDNs place a large number of edge servers (also called *surrogates*) at geographically distributed edge networks, enabling content caching and proximal access for end-users. Thus, how to redirection or route client requests among edge server clusters is a central function of CDNs and its design choice is critical to CDNs. The majority of today's

CDNs leverage the Domain Name System (DNS) as the core of their mapping systems to redirect a client's request into a nearby edge server. Since a DNS-based mapping system is based on real-time measurements of server and network conditions, it can provide fast, accurate, and fine-grained control for request redirection. Thus, its usage has been dominant in leading CDN vendors for operating a large number of edge servers, such as Akamai. However, such a DNS-based mapping system requires DNS records to be very dynamic, which restrains those DNS-based CDNs from authenticating their mapping DNS records by using DNSSEC signatures. This is because DNSSEC was originally designed for static records. Due to its prohibitively high computational overhead, traditional RSA-based DNSSEC is not a feasible solution to secure dynamic DNS records in the context of CDNs.

In this study, we investigate the security implication in the DNS-based mapping system of CDNs, which can be exploited by adversaries to hijack the operation of request redirection in a very stealthy manner and then nullify the benefits provided by CDNs, such as proximal access to nearby edge servers, geographically load-balancing, and DoS protection. To perform the redirection hijacking, the adversaries deliberately maneuver the mapping between end-users and edge servers by injecting fraudulent DNS records. Worrisomely, even the newly adopted Elliptic Curve Digital Signature Algorithm (ECDSA) that is capable of providing real-time DNSSEC signatures is ineffective to detect and prevent malicious DNS record injection. We reveal that an adversary can utilize a legitimate mapping record to override CDN's cluster/server selection and redirect a certain group of users to an edge server chosen by the adversary.

1.4 Roadmap

The remainder of this dissertation is structured as follows. In Chapter 2, we present a large-scale measurement study to profile the characterizations of authoritative DNS (ADNS) servers for modern web services and cloud-hosting subdomains. In Chapter 3, we

propose the domain name based features that are explicitly available from domain name itself to mitigate the performance penalty caused by the prevalent usage of "*disposable domains*" (i.e., one-time-used names) on the Internet by prematurely excluding the domains that are highly unlikely to be used from DNS cache. In Chapter 4, we investigate a fundamental vulnerability of CDNs, called *redirection hijacking*, which stems from the dynamic characteristics of DNS records used for the CDN's request routing, and could be exploited by adversaries for degrading CDN's performance and nullifying the benefits provided by CDN's such as load balancing and DoS protections. We summarize the studies in this dissertation and discuss future work in Chapter 5.

Chapter 2

Measuring and Characterizing the DNS Deployment of Modern Web Services

As a hierarchical distributed database system, the Domain Name System (DNS) is one of the most important components of Internet infrastructure, providing the mapping between the domain names and network-level addresses to direct clients to specific Internet services. In DNS hierarchy, the authoritative DNS (ADNS) servers are administered by the service providers and responsible for storing the name-to-address records and returning answers to the clients. Due to the requirements of extra hardware resources and additional maintenance support, the web service providers are increasingly adopting the upstream authoritative DNS servers. Also, many of today's popular web services are built upon Infrastructure-as-a-Service (IaaS) clouds. In this study, we explore state-of-the-art of the patterns of ADNS deployment and perform a large-scale measurement study to demonstrate the characteristics of different patterns.

In doing so, we first collect the authoritative DNS server information for top-ranking websites on Alexa's list [3] and eliminate the redundant domain records. This constructs our dataset with about 2.3 million nameservers for about 0.94 million websites. We then

develop a systematic method to explore ADNS server deployment patterns and perform the geo-distributed probing experiments. In particular, by directly issuing DNS queries to each ADNS server, we examine their deployment details and characteristics. Next, we focus on the DNS deployment of web services whose subdomains are hosted in cloud infrastructure. We extract the subdomain list from an existing dataset [8], reproduce the ADNS servers of subdomains for comparing with the original results, and examine their deployment.

We summarize our major findings and contributions as follows:

- We use a simple heuristic method to determine the ADNS deployment patterns. In fact, it is fairly easy to recognize the pattern for an individual website from its NS records, but it is much more difficult when looking for millions of websites in such a large-scale study.

- We validate the use of ADNS proxy infrastructure by examining the transition delay and the TTL aging.

- We first quantify the usage and profile the characteristics of ADNS servers in terms of the deployment patterns.

- We find that most top-ranked websites deploy their own DNS servers but emerging popular social sites tend to use the upstream DNS-hosting services. We also observe few servers being used in private deployment.

- We find that the ADNS deployment patterns remain stable. The change of private servers is more frequent than that of upstream servers. The websites using upstream services change frequently their hosting domains but have the lowest frequency to change their deployment patterns.

- Among the studied patterns (i.e., private, upstream, and hybrid), we observe that upstream achieves the highest performance while hybrid has the highest availability.

- We quantify the usage of ADNSes for cloud-hosting subdomains. We observe a noticeable growth on the usage of cloud-providing DNS service.

The remainder of this chapter is organized as follows. We introduce the background of DNS and ADNS deployment in §2.1. We describe the data sets used in the study and our analysis methods in §2.2. We present the measurement results and analysis of ADNS deployment for top-ranking websites in §2.3. We profile the usage of ADNSes for cloud-hosting subdomains in §2.4. We survey related work in §2.5, and finally conclude the chapter in §2.6.

2.1 Background

In this section, we give an overview of DNS and present the authoritative DNS deployment patterns for modern web services. In addition, we specially discuss the DNS deployment of cloud-hosting subdomains.

2.1.1 DNS Overview

Figure 2.1 shows the DNS components and the process of name resolution. A resolution routine on the client-end host, called *stub resolver*, issues a DNS lookup to a *recursive resolver*, a local DNS server (LDNS) deployed by the client's local network or a public DNS service [21, 28] located in a wide area network. Without considering the cache effects on the resolvers and intermediate servers, the recursive resolver will first contact the root server. The root server directs the resolver to query a top-level-domain (TLD) server (e.g., the .com TLD server). Similarly, the TLD server responds the resolver's query with the address of the *authoritative* DNS server (ADNS) for the corresponding domain. Next, the local recursive resolver queries the ADNS server for the address of the domain host, and finally the client can reach the Internet service as the recursive resolver returns the answer for name resolution.

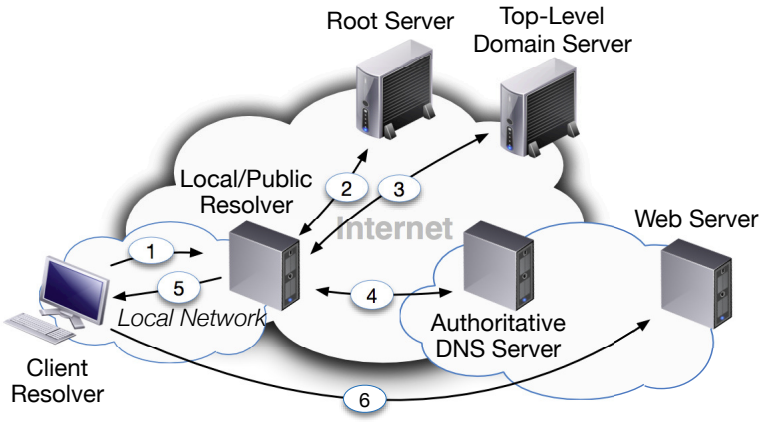

Figure 2.1: DNS Resolution Process (Iterative query).

2.1.2 ADNS Deployment Patterns

Figure 2.2 illustrates the steps of a client accessing the web services under three different ADNS deployments:

- **Private ADNS server:** The web service owners deploy their private authoritative DNS servers only within their own domains.[1]
- **Upstream ADNS server:** The web service owners delegate their authoritative name resolution to the upstream DNS-hosting service providers.[2]
- **Hybrid ADNS deployment:** The web service owners employ both the private DNS servers and the upstream ADNS servers for their authoritative name resolution.

Table 2.1 lists the domains hosting authoritative DNS servers for the top 15 websites on Alexa's list [3]. Most of these top websites host the ADNS servers within their own domains

[1]The domains hosting web services and private nameservers may also be located inside IaaS clouds. In such a case, the service provider runs the ADNS servers with cloud instances.

[2]We only consider the ADNS-hosting domains to identify the deployment, regardless of whether a website itself is hosted in private infrastructure or web-hosting companies.

[3]The ranking is from April 2015.

[4]The TLDs of Ultradns serving for `amazon.com` include `.net`, `.org`, `.info`, and `.co.uk`. Although Amazon offers a public DNS-hosting service (Route 53 [5]) for its cloud tenants, it delegates its DNS resolution to upstream providers. We infer that it is a historical reason: Amazon has been running the upstream ADNS for `amazon.com` since its establishment in 1995 and did not switch to private servers when expanding its business to cloud services.

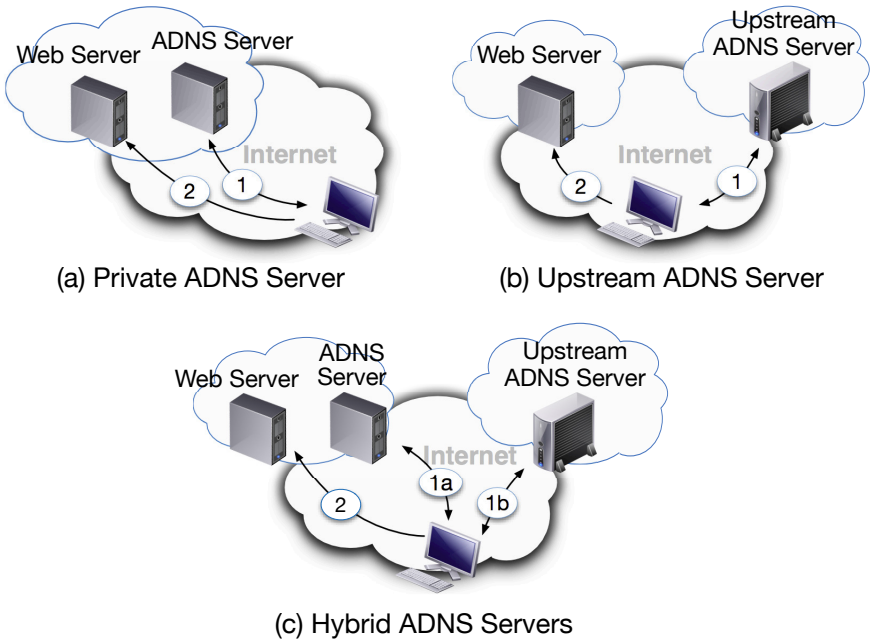

Figure 2.2: ADNS Deployment for Web Services.

since they are capable of maintaining a secure and reliable DNS infrastructure. However, `amazon.com` and `twitter.com` delegate their name resolution services to upstream ADNS providers, for which `amazon.com` uses two different vendors. `Linkedin.com` hosts ADNS servers in both its own domain and the upstream provider. Note that many top websites enable their primary ADNS servers to resolve the names for other domains they possess, such as Google for `youtube.com`, Microsoft for `live.com`, and Sina for `weibo.com`.

Use of DNS Proxy. To validate DNS traffic and protect the ADNS servers, the service owners may deploy the DNS proxy servers to control incoming[5] DNS queries and enable flexible management. Figure 2.3 shows the DNS proxy deployment for private and upstream ADNS servers. In both scenarios, the clients first contact the DNS proxy servers, and then the proxy servers transparently relay the queries to ADNS servers and

[5]The DNS proxy can also be used to control the outgoing DNS traffic in local networks. In addition, a specified re-routing service, called *Smart DNS proxy*, directs the users to access region-restricted or blocked content. In our study, we only consider the DNS proxy that serves for incoming connections of ADNS servers.

Table 2.1: ADNS Deployment of Top 15 Sites[3]

Domain	ADNS
google.com	google.com
facebook.com	facebook.com
youtube.com	google.com
yahoo.com	yahoo.com
baidu.com	baidu.com
wikipedia.org	wikimedia.org
amazon.com	dynect.net, ultradns[4]
twitter.com	dynect.net
taobao.com	taobao.com
qq.com	qq.com
google.co.in	google.com
live.com	msft.net
sina.com.cn	sina.com.cn
linkedin.com	dynect.net, linkedin.com
weibo.com	sina.com.cn

return the answers back to clients. To date the DNS proxy servers have been developed as functionality-rich systems, such as Global Traffic Manager (GTM) [20] or Global Server Load Balancing (GSLB) [27], to optimize access performance and secure DNS servers. All these environments are recognized as the DNS proxy infrastructure in our study.

DNS for Cloud-hosting Subdomains. Figure 2.4 shows the steps for accessing the (partly) cloud-hosting subdomains. These subdomains are the partial sections of primary websites, hosted inside a cloud for achieving scalable infrastructure and providing extended services. The primary ADNS servers, deployed by either private servers or upstream services, could be used to direct users to the cloud subdomain, as shown in Figure 4(a). Also, some providers delegate the name resolution for cloud subdomains to a dedicated subdomain DNS server, as shown in Figure 4(b). The delegated DNS server could be deployed by (1) DNS software running within cloud instances, (2) DNS resolution service

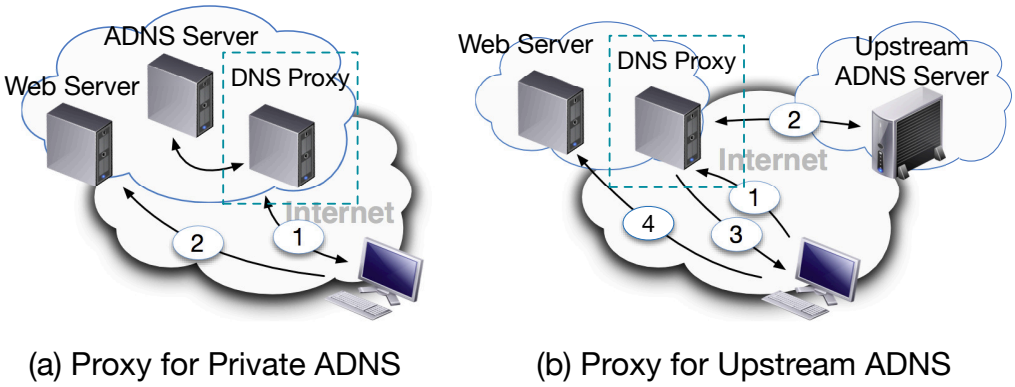

Figure 2.3: DNS Proxy.

offered by the cloud provider, or (3) the third-party upstream services. We examine the DNS deployment details for such cloud-hosting subdomains in §5.

2.2 Datasets and Methodology

This section introduces the datasets used in the study and presents our approaches to examine the deployment patterns of ADNS servers.

2.2.1 Datasets

Alexa's list [3] of the top 1-million websites is updated daily and based on the one-month average traffic. The list used in our study was downloaded in June 2014.

2.2.1.1 Authoritative DNS server dataset

We collected the authoritative DNS server information for the top 1-million websites on Alexa's list through `dig` utility. Since the personal pages from several popular web services are individually included in Alexa's list and their hosting domains have been ranked, we eliminated those page links from our data with (1) a slash "/" after domain names, which typically indicates the sub-webpages, such as YouTube's user home pages `youtube.com/[user]`, and (2) the domains having the lower-level names before the

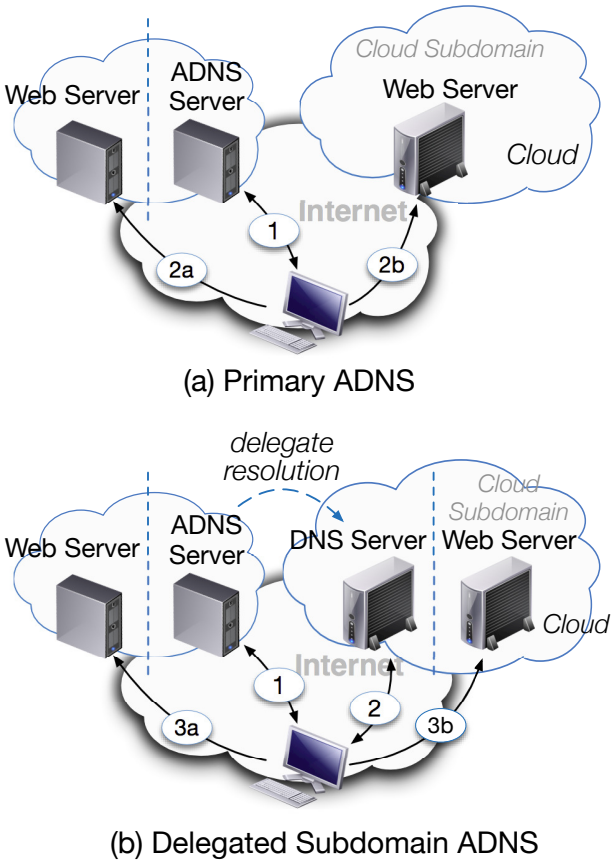

Figure 2.4: DNS Deployment for (Partly) Cloud-hosting Subdomains.

second-level names[6], such as Blogspot's personal blog pages `[user].blogspot.com`. Finally, we collected the authoritative DNS information for 942,467 websites with 2,339,345 domain servers, consisting of 352,022 distinct nameservers.

Upon the dataset, we study the characteristics, such as performance and availability, of different ADNS deployment patterns by probing the collected servers. To eliminate the cache effect at intermediate servers, we issue the DNS queries to designated ADNS servers by using the `@global-server` option of `dig` utility.

[6]They are identified by `publicsuffix`[32], a parser implemented by recognizing the domain suffixes from the Public Suffix List [31].

2.2.1.2 Cloud-hosting subdomain list

The list of cloud-hosting subdomains is extracted from a prior dataset [8], which includes the subdomains associated with Alexa's list and hosted in Amazon EC2 and Windows Azure. We reproduced the DNS server list since we found that a large number of DNS records have been changed. We also found many subdomains had been migrated to other cloud providers, but their DNS information could still be used for our study.

2.2.2 Determining ADNS Deployment Patterns

We first construct the list of authoritative DNS servers of each web domain by extracting `NS` records from the responses of `dig` probes, and then attempt to determine the ADNS deployment pattern. However, without a complete list of global upstream DNS providers to identify all DNS-hosting domains, it is impossible to have an automated method to accurately determine the pattern for every site. To ensure our study's effectiveness and accuracy, we design a heuristic method to capture *as many websites as possible for each pattern* by discarding the records likely to be miscategorized:

Step 1: First we extract the second-level name from each domain (e.g., `google` from `google.com`), and perform a substring search for that name in its every name-server's hostname. Here we exclude the domains with one or two characters of second-level names since they may *coincidently* match the hostnames (e.g., the second-level name "`t`" from `t.co` matches the servers at `dynect.net`). We then move these domains into a `short-name` list to be determined later (Step 5).

Step 2: For each domain, we assign two variables, t_m and t_n, to record the number of matching and non-matching occurrences, respectively.

- If a certain server contributes t_m, we consider that it serves as a private nameserver located in the same or related domain with its web service. We collect those servers in a `private` list.

- For the servers without matching occurrence, we put them to an `upstream` list.
- For a domain with $t_m > 0$ and $t_n = 0$, we consider all its ADNS servers to be located in a related domain hosting its web service, and thus we refer to the domain as the private ADNS deployment.

For example, four nameservers `ns[1-4].google.com` serve for `google.com`. The second-level name `google` matches for all nameservers, which gives $t_m = 4$ and $t_n = 0$. Also, upon the general naming customs, this matching process has been able to recognize many domains that deploy private ADNS servers in separated and dedicated domains (e.g., `ebay.com` deploys its DNS servers in `ebaydns.com`).

However, some separated DNS-hosting domains cannot be identified from the simple substring search in this step (e.g., as shown in Table 1, the nameservers hosted in `sina.com.cn` serve for `weibo.com`).

Step 3: Our basic idea, to recognize such deployment, is to determine the deployment patterns of websites by the categories of their nameservers. Therefore, we would filter the `private` and `upstream` list (in this and the next step, respectively) to exclude the nameservers that might be miscategorized.

Despite the successful matching in step 1, the `private` list still includes some servers used as upstream services. For example, the nameserver `ns1.dnsmadeeasy.com` matches the domain `dnsmadeeasy.com`, and thus is recognized as a private DNS server. However, it also serves as an ADNS-hosting server for thousands of other websites.[7] Since the nameservers will be used to determine deployment patterns, we would eliminate such records from the `private` list.

In doing so, we first extract the domain part of each ADNS server by using the `publicsuffix` [32] parser (e.g., getting the domain part `bbc.co.uk` from the nameserver

[7] In our study, the pattern of servers is determined by their specific role for an individual website. That is, in this example, `ns1.dnsmadeeasy.com` is a private DNS server for `dnsmadeeasy.com`, but an upstream server for other domains using its DNS-hosting service.

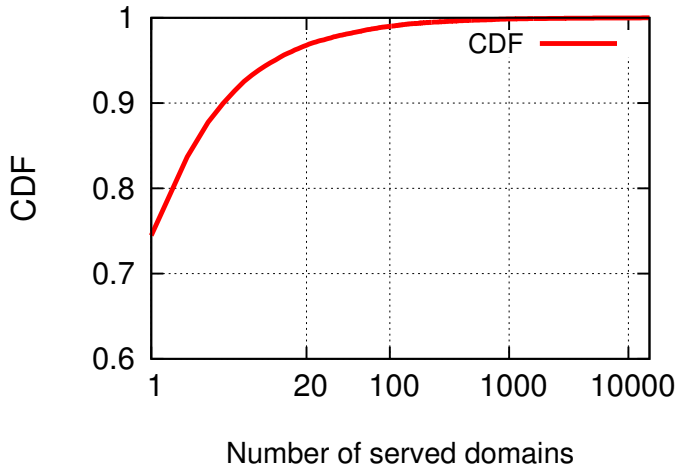

Figure 2.5: CDF for the number of domains supported by individual DNS server under matching occurrence (*Note that the Y axis starts from 0.6*).

`ns1.tcams.bbc.co.uk`). We then calculate the number of domains that each extracted domain part serves for, except for the domains with matching occurrences. For example, the ADNS servers in `ebaydns.com` support 64 domains in the dataset, 49 of which match the hostnames of servers (i.e., the second-level name `ebay` matches the domain `ebaydns.com`). This indicates the ADNS servers in `ebaydns.com` serve for 15 domains without matching occurrences. On the other hand, we find 8,487 domains supported by the servers located inside `dnsmadeeasy.com`.

The heuristic to filter the `private` list is from a general observation: *the number of domains served by private DNS servers would be less than the number of domains supported by the third-party upstream servers*. Since what we want is to have a private server list with accurate classification, simply discarding the servers with a certain number of served domains will be enough for our study.

Figure 2.5 demonstrates the cumulative distribution for the number of domains served by nameservers with *matching* occurrences. Not surprisingly, most of servers only support a few domains: about 96.7% of DNS-hosting domains serve for fewer than 20 web domains. We believe that it is safe to discard the servers administrating more than 20 domains in the `private` list, since (1) the matching process has excluded the majority of upstream

servers, (2) most upstream service providers should have more than 20 served websites in such a large-scale dataset, and (3) while several errors exist, they would not have a significant effect on our study because the number of miscategorized websites is marginal ($< 20 \times$ *the number of errors*).

To inspect the accuracy of our heuristic method, we extract and examine the results for the top 100 most popular websites. We find that the nameservers from three private domains (`google.com`, `facebook.com`, and `msft.net`) are removed from the `private` list since they serve for more than 20 domains. We argue that it also does not affect our study because (1) the ADNS deployment patterns of their primary websites have been identified as private due to the matching occurrence in step 2, and (2) only a small portion of the domains served by these private nameservers in the three domains above are discarded but most of them have the same ADNS deployment as their primary sites.[8]

According to the filtered `private` list, if any server in the `upstream` list also appears in the `private` list, we consider that it is deployed as a private server for the corresponding domain. We then move this server into the `private` list.

Step 4: We extract all domain parts from the servers in the `upstream` list and calculate how many websites those DNS-hosting domains support. For similar considerations with step 3, we discard the servers with fewer than 20 administrated domains since they could be private servers. We also will not move them to the `private` list because without any matching occurrence, we cannot ensure that they are private servers.

Step 5: We then update t_m and t_n for each unidentified domain (i.e., a domain not labeled in step 2 and in the `short-name` list) by examining its nameservers:

- if one nameserver appears in the `private` list, increase t_m by one;
- if one nameserver appears in the `upstream` list, increase t_n by one.

[8]The nameservers of these sites are moved back to the `private` list in the following steps.

We finally determine the ADNS pattern as follows:

- $t_m > 0$ and $t_n = 0$: *private ADNS*
- $t_m = 0$ and $t_n > 0$: *upstream ADNS*
- $t_m > 0$ and $t_n > 0$: *hybrid ADNS*

As an example, we find six nameservers for `hao123.com`, a website-directory service provided by `baidu.com`. One nameserver is located in `hao123.com` and the other five are in `baidu.com`. Thereby, the matching step gives $t_m = 1$ and $t_n = 5$. In addition, the five nameservers hosted in `baidu.com` appear in the `private` list since they are also serving `baidu.com`. Finally, we have $t_m = 6$ and $t_n = 0$, and refer to this site as a private ADNS deployment.

We eliminate the domains whose nameservers cannot be identified by the `private` or `upstream` lists. Finally, we discard 19,956 (2.1%) records from the original dataset, of which 922,489 websites remain to be studied in our measurement. We understand our heuristics, filtering-based method may still not be perfect, but we believe that the method's accuracy is high enough for performing a large-scale measurement, with few errors at an acceptable level.

2.2.3 Validating the ADNS proxy

The use of the DNS proxy conceals the ADNS infrastructure and the proxy discovery is a challenging problem since the DNS proxy exhibits the same behavior as a private ADNS server from the viewpoint of external clients. We perform two different probing tests to study the proxy infrastructure.

We first discover the usage of a proxy by roughly estimating the DNS response latency within authoritative resolving infrastructure: if a DNS query undergoes a distinctly longer response delay than a *ping* probe, the website has a very high probability of using a proxy. We probe 138,240 private servers for 70,502 domains from three vantage points (at eastern-, middle-, and western-US), and we observe that 73% of servers respond to the *ping* probes.

Then, for each of those servers, we attempt to identify and normalize the time difference in response latency between *ping* probes and DNS queries. The normalized difference in response latency (i.e., round-trip-time, RTT) between DNS and *ping* is computed as below:

$$(RTT_{\mathbf{dns}} - RTT_{\mathbf{ping}})/RTT_{\mathbf{dns}}$$

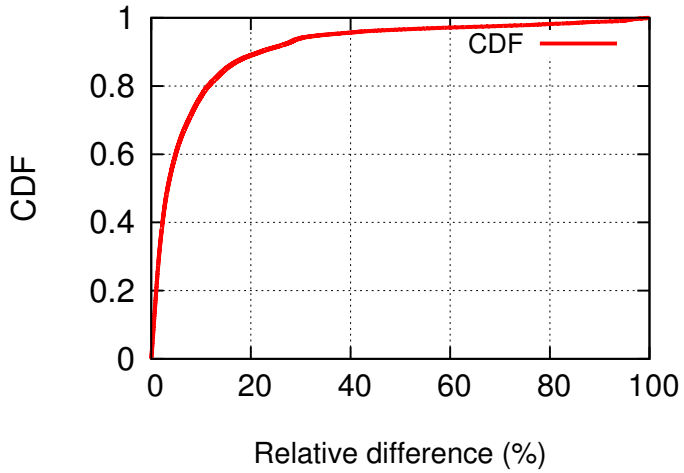

Figure 2.6: CDF for the difference of responses by *ping* and *dig*.

We measure the normalized difference values in three vantage points and define their average as the relative difference for each server. Figure 2.6 plots the distribution of relative difference. We observe that (1) the majority of servers (close 90%) whose relative difference values are less than 20%, but (2) indeed there are 3.4% and 1.8% of servers whose relative difference values are higher than 50% and 80%, respectively, i.e., their DNS queries have response times 50% and 80% higher than the RTTs of *ping* probes. Assuming the same or at least similar routing paths for consecutive *ping* probes and DNS queries, even if we cannot accurately identify the proxy for each website, our probing results clearly indicate that the use of ADNS proxy infrastructure does exist but the usage is very limited.

If a proxy enables the cache, we cannot use the relative difference to detect the use of a DNS proxy anymore, since the cache will respond to the queries and no significant additional delay would be noticed. However, different from an ADNS direct response, in

which TTL is a fixed default value, the TTL value of a proxy cached record decreases with time elapse. This is because the cache at a DNS proxy will age the cached records like a local resolver. That is, the answers from a proxy will have reduced TTLs. Moreover, the time resolution in TTL is in seconds. Therefore, we send two successive queries with an interval of 10 seconds to these ADNS addresses and then use the reduced TTL values to detect the existence of a DNS proxy. In other words, if we can detect that the difference of TTLs between two successive DNS queries is round 10s, a proxy infrastructure may have been adopted to conceal the actual ADNS servers. However, by probing 70,520 domains with private deployment, we only find 75 (0.001%) domains where TTL-reduced records from their ADNS servers occur. This observation is consistent with the proxy detection result above, i.e., the ADNS proxy infrastructure is not used by most service owners. Furthermore, comparing the two results (3.4% vs 0.001%), we can see that most ADNS proxy servers simply relay the authoritative records to clients and no cache is used.

2.3 Measurement Results

This section presents the measurement results and analysis for the websites on Alexa's list. First we quantify the usage of deployment patterns and examine the fundamental deployment configurations, such as the number of servers and TTLs of DNS records. Then we present the performance study based on the probes from the geo-distributed locations and analyze the availability for different patterns.

2.3.1 Overview

2.3.1.1 Deployment Patterns

Table 2.2 summarizes the pattern recognition in our dataset, based on the breakdown by the top 100, 1,000, 10,000, and all records. The majority of websites delegate their authoritative name resolution to upstream DNS-hosting services for the simple management. The high-ranking sites are much more likely to deploy their own ADNS servers:

the fraction of private deployment decreases sharply as the number of studied domains increases. In addition, the percentage of hybrid-deploying domains remains stable on about 4-6% within top 10,000 websites but overall only 1.4% of websites are recognized as such a deployment. Table 2.3 lists the top 10 web domains of each deployment pattern. Many emerging top websites, especially the social sites popular on mobile web, such as Twitter, Pinterest, and Tumblr, use the upstream services to facilitate the quick and convenient deployment.

Table 2.2: Summary of Deployment Patterns

Patterns	**Top 100**	**Top 1k**	**Top 10k**	**All**	
	%	%	%	#	%
Private	68	34.1	18.76	70,520	7.6
Upstream	27	60.2	77.17	838,605	91.0
Hybrid	5	5.7	4.07	13,364	1.4
Total	100	100	100	922,489	100

Table 2.3: Top 10 sites for each deployment pattern

Private		**Upstream**		**Hybrid**	
rk.	**Domain**	**rk.**	**Domain**	**rk.**	**Domain**
1	google.com	9	amazon.com	15	linkedin.com
2	facebook.com	12	twitter.com	20	ebay.com
3	youtube.com	30	pinterest.com	65	cnn.com
4	yahoo.com	34	tumblr.com	84	ebay.de
5	baidu.com	38	paypal.com	99	ebay.co.uk
6	wikipedia.org	39	instagram.com	115	nytimes.com
7	qq.com	42	xvideos.com	120	pixnet.net
8	taobao.com	45	imdb.com	148	livedoor.com
10	live.com	48	ifeng.com	167	skype.com
11	sina.com.cn	49	amazon.co.jp	171	ups.com

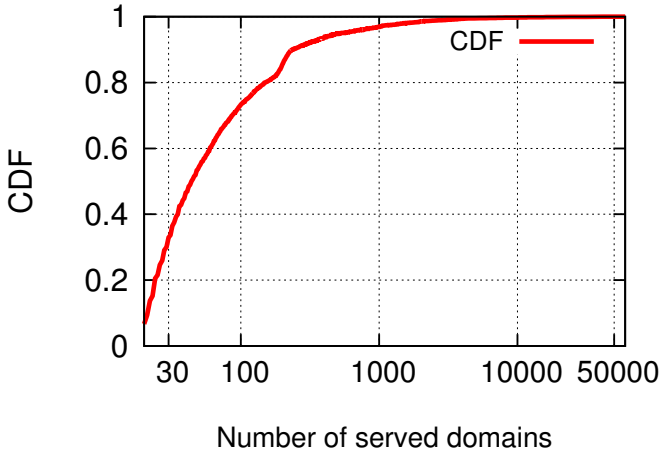

Figure 2.7: CDF for served domains of DNS-hosting domains.

2.3.1.2 DNS-hosting Providers

Here we examine the upstream DNS-hosting providers profiled by our dataset. Figure 2.7 shows the distribution of the number of served domains for extracted DNS-hosting domains. The popularity of upstream domains is quite skewed. We observe that about 70% of hosting domains support fewer than 100 websites, and only 1.4% of hosting domains serve websites more than 1,000.

Table 2.4 lists the top 10 DNS-hosting domains in terms of the number of websites they support in Alexa's list, with the number of servers identified in our dataset. These top 10 domains serve for 26.7% of websites using upstream ADNS. If we include all the domains with more than 1,000 served sites (i.e., 1.4% of hosting domains), this proportion increases to 62.6%. Meanwhile, as the second column of Table IV shows, the quantities of DNS-hosting servers vary considerably,[9] from a few to thousands. This implies that the DNS-hosting providers employ different system designs and implementations to achieve the load-balancing and reliable upstream ADNS services.

Note that the numbers of websites in Table 2.4 are produced by extracting hosting

[9]These servers are identified by their hostnames. In some cases, the service providers prefix the clients' domains to their nameservers to form the client-specific hostnames of ADNSes. This causes the over-estimation of the number of servers for some providers (but not a common case).

Table 2.4: Top DNS-hosting Domains

Domain	**# of servers**	**# of websites**
domaincontrol.com	96	61,455
cloudflare.com	107	27,821
hostgator.com	4,913	22,483
ovh.net	1,960	14,004
bluehost.com	6	12,307
worldnic.com	100	10,855
dnsmadeeasy.com	15	8,487
dnspod.net	12	8,404
name-services.com	13	8,338
dreamhost.com	5	7,593

domains, not the service providers. In fact, several providers offer hosting services through multiple domains, e.g., Ultradns (see footnote 4), and Amazon's DNS service. Amazon deploys a set of hosting domains (256 found), named `awsdns-××.com/net/org/co.uk`, among which the quantities of served web domains remain balanced and stable (i.e., 220-320).

2.3.2 Number of Nameservers

We quantify the usage of ADNS servers in terms of the deployment patterns. Figure 2.8 plots the cumulative distribution of the number of ADNS servers for each pattern.[10]

Generally, the websites using private ADNS servers tend to deploy slightly fewer servers than the sites using upstream services. This indicates that those sites have the ability to maintain a reliable DNS infrastructure, and various back-end techniques may be used, such as load balancing and failover. On the other hand, simply using more upstream servers is a convenient practice to achieve reliable name resolution. In addition, when looking at the quantities in terms of ranks, as shown in Figure 2.9, we observe that the high-ranked

[10]We omit the plotting for the overall result since it would show a similar distribution with the upstream pattern due to its dominant quantity.

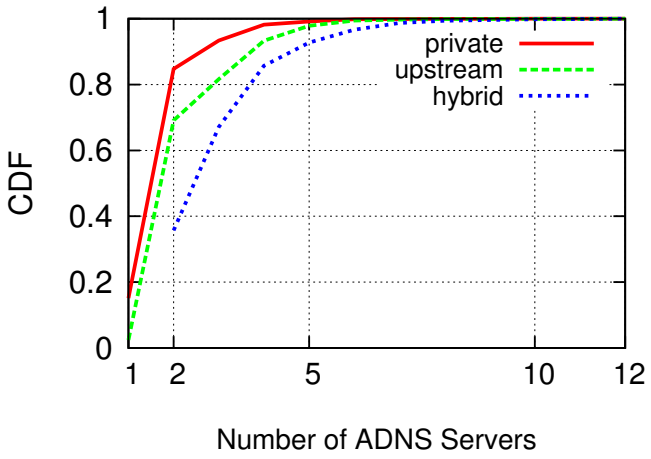

Figure 2.8: CDF for the number of ADNS servers.

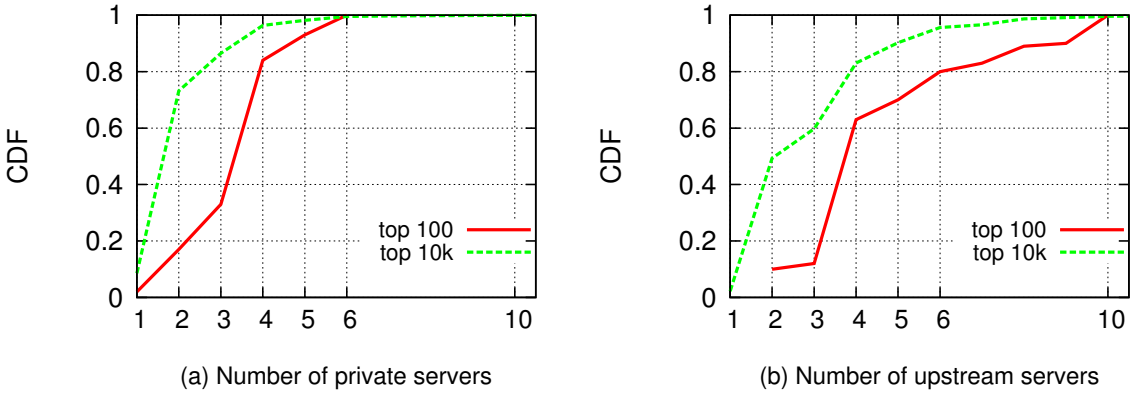

Figure 2.9: CDFs for the number of ADNS servers of top sites (private and upstream).

sites have more servers than others to handle the high-volume accesses. Figure 2.10 shows the CDFs for the numbers of private and upstream servers in hybrid deployment. They exhibit very similar distributions, and the number of servers for either type is less than that in pure private or upstream pattern.

In our dataset, there are only 209 ($< 0.02\%$) and 28 ($< 0.003\%$) domains that have more than 12 and 13 DNS servers, respectively. For easy presentation, we do not show those domains in Figures 8-10. By briefly examining those websites equipped with a large number of ADNS servers, we recognize that 60% of them having more nameservers are mainly for the purpose of *mutual delegation*. That is, a group of websites are served by a

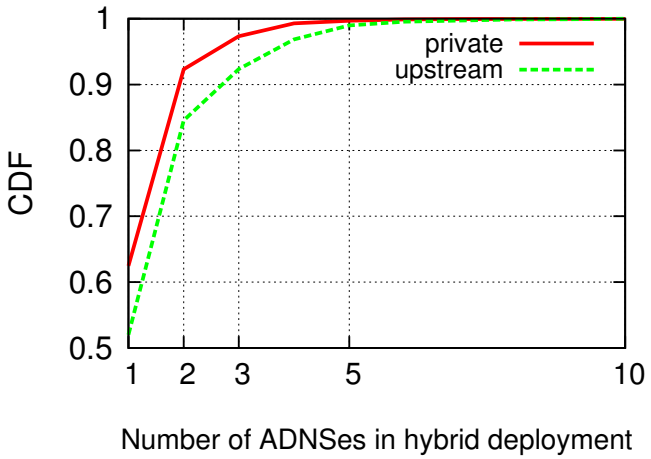

Figure 2.10: CDF for the number of private and upstream servers in hybrid deployment.

set of nameservers that consist of the private servers from every site. These websites with mutual delegation mainly fall into two categories: porn-related and loan-related websites, perhaps for circumventing the local laws or web inspection. This observation implies that the large number of ADNS servers could be a rough indicator of spam websites.

2.3.3 Time-to-Live Values

We now profile the Time-to-Live (TTL) settings of ADNS servers. Figure 2.11(a) shows the cumulative distribution of TTL values of NS records for different ADNS deployment patterns. We observe that the NS records extracted from the nameservers of a hybrid deployment have shorter TTL values than the records of private and upstream patterns. One possible reason to explain why this happens is that: the domain that decides to deploy hybrid DNS servers demands high availability and reliability, thus its system administrators are more meticulous and tend to maintain resilient configuration to quickly respond to the changes on ADNS settings.

Figure 2.11(b) shows the distribution of TTL values of A records. The A records of websites using upstream services have shorter TTL values since the majority of those websites run small businesses and may change their service deployment more frequently. The larger TTL values of websites following the hybrid pattern imply that those websites

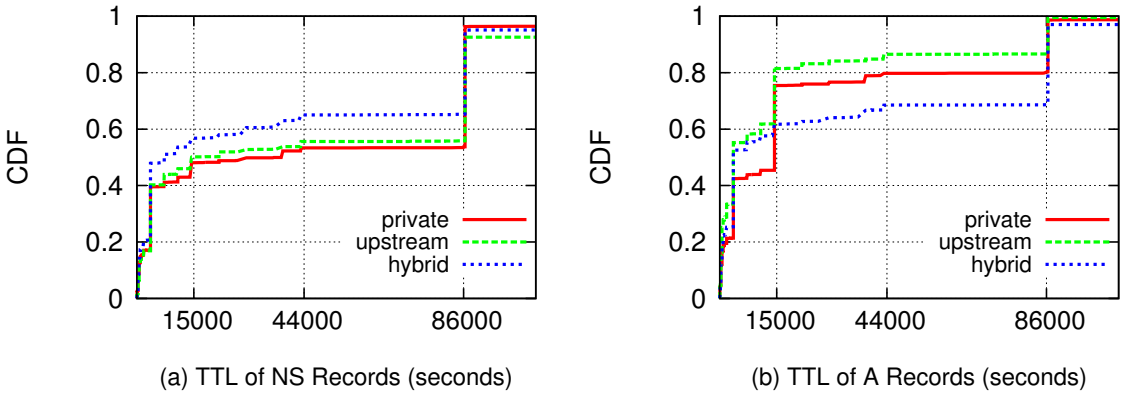

Figure 2.11: CDFs for TTL values.

indeed would like to maintain more stable services.

The larger TTL values of NS records have been identified by many prior studies [79, 65]. However, when comparing Figures 2.11(a) and 2.11(b), we observe that the hybrid deployment pattern has very similar TTL distributions for the NS and A records, while the NS records of private and upstream servers indeed demonstrate the larger TTL values than their A records.

2.3.4 Life-cycle of ADNS Servers

Our probing experiments are performed based on the ADNS server dataset collected in June 2014. After that, we also reproduced two additional lists of ADNS server information in December 2014 and April 2015, for the same Alexa's list in June 2014, to examine how many records have been changed. We observe that 20.4% of ADNS records have been changed in December 2014, and 33.5% of records have been changed in April 2015.[11]

We study the ADNS changes at different levels. We refer to all changes on ADNS records as *server change*, i.e., the nameservers to be added, removed, or relocated. The *domain change* means that the nameservers are relocated to different hosting domains. The *pattern change* means that the websites re-deploy their ADNSes with a different

[11] All the change rates in April 2015 are based on the comparison with the original dataset in June 2014.

infrastructure. Clearly, the domain change indicates the server change, and the pattern change indicates both the server change and the domain change.

The detailed breakdown appears in Table 2.5. Although a large number of ADNS records have been changed, we observe that the deployment patterns remain stable. The websites using upstream DNS services frequently change their hosting domains (i.e., third-party service providers), due to the low cost to migrate their authoritative records; but they also have the lowest frequency to change their deployment patterns, due to the high cost to deploy extra infrastructures to host their ADNS records.

Table 2.5: Summary of ADNS changes (%)

Patterns	**Dec. 2014**			**Apr. 2015**		
	Ser.	**Dom.**	**Pat.**	**Ser.**	**Dom.**	**Pat.**
Private	23.2	4.57	2.90	37.5	7.09	4.51
Upstream	16.7	8.56	1.78	24.7	13.59	2.47
Hybrid	25.3	4.35	3.22	37.2	6.66	4.67
Overall	20.4	6.34	2.53	33.5	8.05	3.68

2.3.5 Performance

To evaluate the DNS performance, we performed the DNS lookups for each nameserver of every domain, which involves 2,223,972 nameservers for 922,489 websites, from 70 globally distributed PlanetLab [30] nodes, and then we clustered the response times into groups according to the deployment patterns. The probings are performed twice each week during a one-month period. Figure 2.12 plots the cumulative distribution of the response times for each pattern. It demonstrates that the upstream services have a small but noticeable performance advantage, since the mainstream DNS-hosting providers have established the distributed infrastructure and spent significant efforts in optimizing their global access.

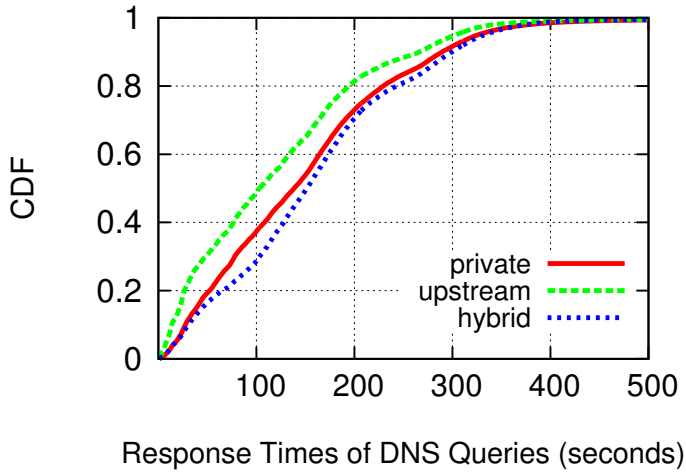

Figure 2.12: CDF for Response Times of DNS Queries.

2.3.6 Availability

To examine the availability of authoritative DNS servers, we first analyze the responsiveness from the probing experiments above. During each probing period, we record a query's response time and retry the unresponsive servers up to three times before we drop the probes. We then calculate the average rates of successful probes. We refer to the results from the performance experiment as Probe 1.

However, the performance probes focus on the response time and do not reflect the availability for a certain period of time. Thus, besides conducting the performance probes, we also perform an active probe experiment with an exponentially distributed interval of a one-hour mean, the same method used in [99], where the availability is defined as the ratio of the number of probes being responded and the number of probes being sent. We select four geo-distributed nodes (located at eastern- and western-US, Europe, and Asia) to issue the probes and run the measurement for approximately five weeks. We refer to the results from the exponential probes as Probe 2.

Table 2.6 summarizes the availability statistics from both probe sets. For each deployment pattern, its server availability refers to the ratio between the successfully responded nameservers and the probed nameservers. The domain availability indicates the chance in

Table 2.6: Summary of Availability (%)

Patterns	Probe 1		Probe 2	
	Server	**Domain**	**Server**	**Domain**
Private	92.54	96.77	95.63	98.65
Upstream	94.44	97.76	98.87	99.04
Hybrid	94.33	97.71	96.93	99.89
Overall	93.54	97.47	97.39	98.98

percentage that at least one of the domain's nameservers responds to the probes. The majority of ADNS servers are available for almost the entire duration. In both probe sets, the servers in upstream deployment have the highest availability since they are more powerful and expensive servers aiming to provide DNS services for many customers. At the domain level, as one would expect, the hybrid deployment indeed exhibits higher availability than the other two deployment patterns in Probe 2.[12]

We observe that a few servers become dead during the experiment of Probe 2. We consider a nameserver becoming dead if the server remains no response for at least one week and until the end of the experiment. The dead servers/domains are removed from the results in Table VI. We observe that 3.3% of servers become dead, 87.4% of which are private servers. We further perform another probing in April 2015 to check the status of those severs that have been identified as dead in Probe 2. We find that only 6.4% of them respond to the probes, which implies that the corresponding domains having responsive servers now may have re-deployed their ADNS servers.

2.4 Cloud-hosting Subdomains

In this section, we explore the DNS deployment for subdomains hosted in IaaS clouds.

[12]In Probe 1, the upstream and hybrid deployment patterns demonstrate similar availability at the domain level, perhaps due to the occurrence of temporary network outage or congestion during the probes.

2.4.1 Deployment

We first extract the cloud-hosting subdomain list from [8] and reproduce the ADNS servers of subdomains in July 2014. Since the subdomains are associated with the websites on Alexa's list,[13] we use our `private` list to identify the private nameservers. Also, due to the simplicity of cloud DNS and dominant quantity of subdomains in Amazon EC2, we only examine the EC2-using subdomains in our study.

Here we split the upstream service providers into two categories: the *cloud-provided* DNS and *non-cloud-provided* services (we still call them upstream providers). The ADNS-hosting servers of Amazon are simply identified by their hostnames, including `amazonaws`, `awsdns`, and `CloudFront`. We also add a new category of ADNS deployments, called *cloud-private*, to represent the private DNS servers running atop EC2 instances. We extract the primary domain parts and perform the search like our previous step in Section 2.3. If we recognize a matching occurrence and the nameservers are not in the `private` list, we check if its IP address is located in Amazon EC2's IP range [4].

Table 2.7 summarizes the usage of the ADNS deployment patterns, showing a different pattern usage. However, it would be similar to Table II if we combine the private and cloud-private, as well as the upstream and cloud-provided, into one category. We also observe a usage growth of cloud-provided DNS deployment from the original dataset (about 54%). Figure 2.13 plots the number of nameservers used by the cloud-hosting subdomains, which is slightly higher than the numbers shown in Figure 2.8.

2.4.2 Subdomain Delegation

Recall that in Figure 2.4(b), the primary websites may delegate the resolution of their cloud-hosting subdomains to dedicated nameservers. We identify the delegation by extracting all subdomains' CNAME records and search the CNAMEs in all NS records. We observe that 4.7% of cloud-hosting subdomains use this deployment style, and 97.6%

[13]The subdomain list [8] was generated by Alexa's list in February 2013. We exclude the subdomains whose primary domains are not with Alexa's list.

Table 2.7: Summary of Deployment patterns for cloud subdomains (%)

Patterns	%
private	11.49
cloud-private	5.40
upstream	15.20
cloud-provided	66.59
hybrid	1.32

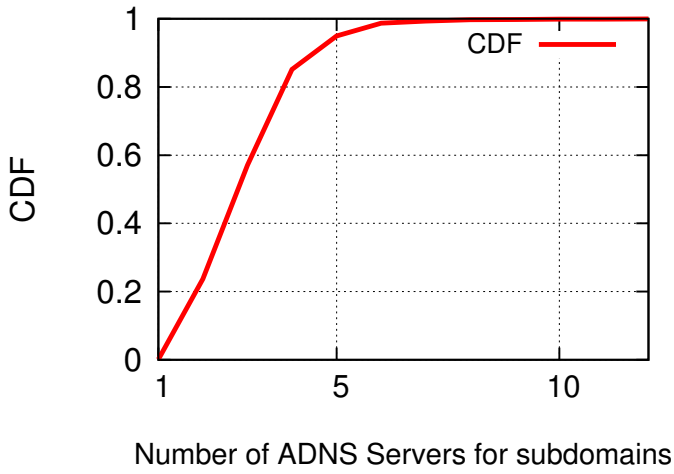

Figure 2.13: CDF for the number of ADNS servers.

of them delegate the subdomains to upstream DNS-hosting providers. Not surprisingly, 89.4% of upstream servers are identified as cloud-provided nameservers.

2.4.3 Life-cycle of Servers

Table 2.8 summarizes the statistics of the changes of DNS records for cloud-hosting subdomains. We recollected the DNS information in April 2015 to compare this dataset with the original one [8] (from February 2013) and the dataset used in our study (from July 2014). We observe a much greater change rate for both nameservers and hosting domains than the identified changes in Table 2.5.

Table 2.8: Summary of DNS changes for cloud subdomains (%)

Change Pattern	Jul. 2014	Apr. 2015
Server	35.49	47.51
Domain	18.27	24.68
Pattern	14.64	20.39

2.5 Related Work

Pang *et al.* [99] presented a comprehensive DNS study by characterizing the properties of local and authoritative DNS infrastructures for the availability, usage, and deployment of DNS. Sisson [19] presented a survey that reports the number of DNS servers on the Internet and various aspects on configuration, such as the recursive support and security configuration. Our work revisits several of their key findings and examines these properties with respect to the evolution of ADNS deployment associated with the use of cloud and upstream resolvers. Schomp *et al.* [107] presented the measurement techniques to discover the client-side DNS infrastructure and studied its behavior on caching. He *et al.* [69] examined how modern web services are using the cloud for deployment of their front ends. Our work focuses on the deployment and characteristics of authoritative DNS infrastructure for cloud-hosting services.

There have been studies to investigate the DNS infrastructure. Gao *et al.* [65] conducted a comprehensive measurement study on global DNS resolvers to reaffirm some findings in previous works and reveal the key differences from root and local perspectives, respectively. Callahan *et al.* [50] passively monitored DNS traffic within a residential network to understand server behaviors and properties of the modern DNS system, such as DNS responses and the violation of TTLs. Jung *et al.* [79] presented a detailed analysis of DNS traces to evaluate the client-perceived performance and the effectiveness of DNS caching, and to simulate the effect of varying TTLs and sharing caches. Liang *et al.* [88] investigated the latency of upper DNS hierarchy and studied the impact of uneven distribution of top-level DNS servers on end-user latency. Pappas *et al.* [102] studied

the reduced availability and increased query delays caused by DNS misconfigurations, and presented three specific widespread types of misconfigurations: lame delegation, diminished server redundancy, and cyclic zone dependency. Ager *et al.* [36] compared the local DNS resolvers against open DNS resolvers, i.e., GoogleDNS and OpenDNS, to examine the latency of DNS resolvers and the content of DNS caches.

There have also been prior works to examine the characteristics of DNS services from various aspects. Liston *et al.* [90] identified the diversity of DNS performance and investigated the degree to which they vary from site to site. Deccio *et al.* [62] proposed a model for server dependencies to measure DNS availability. Castro *et al.* [51] characterized the workload at root servers and analyzed some trends for DNS evolution, such as DNSSEC and DNS IPv6. Berger *et al.* [46] examined the associations between IPv6 addresses and IPv4 addresses of Internet DNS resolvers. Ramasubramanian *et al.* [105] studied the security aspects of nameserver dependencies and delegations. Cranor *et al.* [58] identified the distribution of DNS servers in clusters. Otto *et al.* [97] studied the end-to-end impact of using remote DNS services in CDN, which breaks the assumption that the location of clients' DNS resolvers is close to the actual location of clients.

2.6 Summary

We conduct a large-scale measurement study to quantify the deployment patterns of authoritative DNS servers and examine the characteristics of the patterns. We develop a simple heuristics-based method to determine the ADNS deployment patterns of web domains on Alexa's top 1-million list. We observe that a majority of websites host the ADNSes in upstream services, but the top-ranked sites tend to deploy their own ADNSes. We then perform a probing experiment and observe the performance advantage from upstream services. The hybrid pattern exhibits the high availability due to the redundant deployment. Finally we examine the usage of ADNSes for cloud-hosting subdomains, and observe a noticeable growth in the use of cloud-providing DNS hosting services.

Chapter 3

Exploring Domain Name Based Features on the Effectiveness of DNS Caching

DNS cache plays a critical role in domain name resolution for reduced workloads and low response latency to clients. In this work, we study the negative effect on DNS caching caused by a new class of domain name misuse, the "one-time-use" or disposable domains, and explore and validate the features that could be used to identify this class of misused domain names, without employing complicated deep inspection techniques.

Since most misused domains, either malicious or benign, tend to transmit information over DNS query names, we explore the distinct features from domain name itself that are explicitly available from individual queries. Based on DNS trace logs captured in the resolvers of campus networks, we extract the *re-used* and *once-used* RRs. The reused RRs indicate that the queried domains are retrieved for multiple times, and the once-used RRs only appeared once in one trace. By analyzing a large amount of once-used entries, we observe that several explicit domain name-based features are capable to characterize the reusability of domain names. Based on these features, we propose simple policies to enhance the effectiveness of DNS caching by preliminarily excluding those unreusable RRs.

In order to validate their capability, we quantify the statistical properties of each feature and build a classifier that combines the features. Our classification results demonstrate that our policies are able to prevent around 85% once-used RRs from being cached while only less than 1% reusable RRs are mistakenly kept out of the cache.

The reminder of this chapter is organized as follows. We introduce the background of DNS caching and disposable domains in §3.1. We present the proposed domain name-based features on DNS caching in §3.2. We analyze the collected datasets and build a classifier to validate our features in §3.3, and conduct a trace-driven evaluation in §3.4. We survey related work in §3.5. Finally, we conclude in §3.6.

3.1 Background

3.1.1 DNS Caching

Recursive DNS resolvers retrieve the name resolution results for clients and cache the received responses to answer the following queries. The duration that the cached records would be valid is specified by a *time-to-live* (TTL) value.

In standard TTL-based caching, the TTL value is set and handed out by the administrator of *authoritative* DNS record, and the cached entries are expunged after their TTLs expire. The duration for caching a negative response (e.g., NXDOMAIN, NODATA, *etc.*) is given by the TTL value of SOA record [38]. However, although the TTL-aging-based behavior is legible, the violation of TTL is observed pervasively both in modern web browsers and DNS infrastructures [50]. For instance, many browsers and resolvers assign a minimum amount of seconds for holding a RR, and many resolvers trim the large TTL values to a default maximum value.

3.1.2 Domain Name Misuses

The use of DNS in various ways for which it was not originally designed has been observed for many years. For example, DNS is exploited as an effective covert channel for surrepti-

tious communications [103, 127]. Moreover, Chen *et al.* [54] studied *disposable domains*, a more generic class of domain misuse where the query names are adopted to convey the "one-time signals". These domains are not necessarily malicious and are observed pervasively from various types of service providers, including popular search engines, social networks, CDNs, and security companies, and have being increased to a significant portion of queried domains on the Internet. Due to the "one-time-use" pattern and the increasing use of such domains, the DNS cache would be filled with entries with *near-zero* hit rates. Our work is mainly built on their analysis for disposable domains.

3.2 Domain Name-Based Features

	F1	F2	F3	F4
0.28e.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.27ebb55310eb3785446c4874fefabd756df2 ↪ ff23612f512d4ddc7c9a3ba70d2.b.f.01.s.sophosxl.net	•	▲	★	▼
f6a9fc3efdffd146663f44de1c071c0f548c5653.p.00.s.sophosxl.net		▲		
p4-hnpieeqf4ghwk-ab6awvsfyjixzdw7-629649-i1-v6exp3-v4.metric.gstatic.com		▲		▼
0107c2e22801.t-1436279541.i45381316.04e6d5fe76b6755a1edd7532a34ec877-27718-htm. ↪ fp.bl.barracudabrts.com	•	▲	★	▼
f6a556b42904cfd118c0-553848320f40ae46bf95fbb566795773.r63.cf2.rackcdn.com	•	▲		
ada1a1b36908553d09b507630a84f2606.profile.lhr5.cloudfront.net		▲		
e4cbe5a2594fa1dd8306275ef1e7e4df.azr.msnetworkanalytics.testanalytics.net	•	▲		
b-0.19-a3000008.8011081.1644.981.3ea3.410.0.q3j4p1csa3z1seadcmamni9295.avts. ↪ mcafee.com	•		★	▼
0.0.0.0.1.0.0.4E.c7eijlj4gwumadva92s62m52ri.avqs.mcafee.com			★	▼
i1-j1-18-15-4-114-3425533130-i.init.cedexis-radar.net		▲	★	▼

Figure 3.1: Sample of domains with the domain name-based features

Domain names are human-readable and easy-to-remember character sets. However, the once-used domains exploit the query names as a communication channel. One of our insights is that such misused domains are encoded automatically to convey formatted information and should have significantly different patterns on their names from normal domain names.

Therefore, we consider the possibility of characterizing the once-used domains, and then exploit the derived features to filter out the disposable domains. The removal of such once-used domains from the DNS cache will improve its performance, because the

pervasive use of misused domains has the DNS cache occupied by those entries that are highly unlikely to be reused. Figure 3.1 presents the preliminary examples that motivate our feature selection. We simply plot the distributions of the query name length and the number of subdomains (i.e., the subdomain depth), respectively, for *all* observed query names and *distinct* domain names from one of our trace logs (§3.3). It is evident that (1) most repeatedly appeared domains have short name and limited subdomain depth and (2) a significant portion of domains have long query name and a large number of subdomains. This implies that, under a limited memory space, discarding those entries with long name or deep subdomain as they are scarcely to be retrieved again, would save more space in the cache to store the entries with higher possibility of being re-used, and thus effectively improve the caching effectiveness.

Based on the analysis of large amounts of once-used domains, we identified that the domain name-based features, such as the two features above, are able to characterize the caching behavior of domains, without the help of sophisticated features used in detection of malicious domains (e.g., the behavioral features in EXPOSURE [47] and statistical features in Notos [39]). As a result, we propose the following features and explain why they may affect the caching effectiveness. In spite of straightforwardness, all but the first one have not yet been characterized in the analysis of DNS and none of them has been studied in context of caching performance.

- `F1`: **Length of query name**: Since most of once-used domains tend to send messages over DNS queries, those domains naturally have (much) longer query names to pack as much information as possible and would hardly be reusable. Thus, we consider that there could be a strong correlation between the length of query names and once-used RRs in the cache.

- `F2`: **Length of the longest-subdomain name**: Similar to the query name, the individual lower-level name, i.e., the string representing a subdomain, could also be larger than a legitimate subdomain name that tends to be "easy-to-remember".

- `F3-a`: **Subdomain depth**: To report information over DNS, the implications of domain names have to be easily recognized by the receivers. In doing so, the domain separator, i.e., the period "**.**", would naturally be employed to format the domains to give the name strings meaningful information, resulting in a deep subdomain level, i.e., a large number of subdomains in one query name.

- `F3-b`: **Number of format fields**: Like the period specifying the subdomain level, we also observed that the hyphen "**-**" is widely used as field separator to format the messages in one subdomain name.

Due to the similarity on their function, in our study we treat both the periods (`F3-a`) and hyphens (`F3-b`) as equal format separators and employ the same term, "number of format fields" (`F3`), to represent the number of strings separated by either "**.**" or "**-**".

- `F4`: **Number of fields with unusual lengths**: To represent various pre-defined types of information inserted in specified positions, the length of format strings would vary widely and many of the fields would be either unusually long or unusually short. We consider that such domains are also quite hard to be reused. Thus, we define a metric of the sum of the number of long-format-field (L-FF) and short-format-field (S-FF) within one query name to identify such a feature.

Figure 3.2 shows the sample of domains with the explicit features. Note that these features, e.g., the length of query names, do not necessarily indicate the malicious purposes [103] (actually most of them are benign). However, they indeed indicate the usage of "signaling" and thus imply the high possibility of "one-time-use" pattern. Thus, there would be a strong correlation between each feature above and once-used RRs in the cache. By exploiting these features, we can revise the caching policies to proactively prevent those RRs that are less likely to be reused from being cached, such that the effectiveness of DNS caching could be significantly improved.

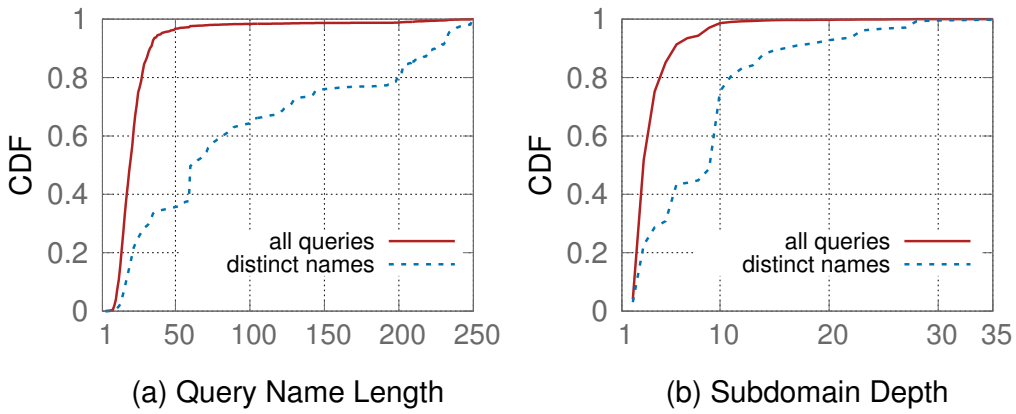

Figure 3.2: Example of distribution for lengths and depths

Methodology. For the proposed features, we characterize the properties of re-used and once-used domains, train a classifier to classify the entries, and conduct a trace-driven simulation to validate their efficacy on caching. In the feature validation (§4.2) and classification (§4.3), the analysis simply relies on the domain names and assumes implicitly that both the cache size and TTL values are unlimited. This assumption simply creates an ideal scenario for caching RRs, and cache hits are not limited by the cache size and TTL values. The simulation (§5.2) runs within a resolver program that caches the entries according to the classification results and common practices on modern DNS resolvers.

3.3 Measurement Analysis

3.3.1 Dataset

The datasets used in the study are the trace logs of outgoing DNS queries over a period of two weeks at local DNS servers in College of William and Mary (`WM`) and University of Delaware (`UD`). We summarize the datasets in Table 3.1.

The trace logs from campus vantage points have two limitations. (1) The traces may not reflect the dynamics of domain names observed at ISP's DNS servers. However, due to the similar patterns of disposable domains from a large-scale ISP dataset reported in [54],

Table 3.1: Summary of datasets

Dataset	Dates	# Records	# Distinct name	# Reused	# Un-reused	Features†
WM	6/25/2015 - 7/8/2015	192,251,799	4,470,732	619,045	3,851,687	N, S, K, T
UD	12/8/2015 - 12/23/2015	1,011,877,341	13,179,395	2,852,021	10,327,374	S‡, T

† Features: Query Name (**N**), Structure of Domain Name (**S**), Query Type (**K**), and Query Timestamp (**T**)
‡ In UD dataset, the actual query names are anonymized but remain distinguishable to identify reappearance. Meanwhile, the structure of domains is given by the length of each format field (e.g., a.b.c-d-e.f.g, where each letter represents the length of each field).

we believe that the proposed policies would be also effective in ISP's DNS cache, especially due to the heavy load on their resolvers. Also, we removed the RR entries that retrieve the local domains. (2) The length of our trace logs is limited. We believe, however, it is still capable to demonstrate the typical cache usage patterns of local resolvers since the origin TTL values of A/AAAA records are typically shorter than one day [65].

3.3.2 Feature Validation

Given the heuristics presented in §3.2, we validate our speculation that these explicit domain name-based features will help to improve the effectiveness of DNS caching. From Table 3.1, we can see that 86.15% and 78.63% of queried domains appeared only once in each dataset, which are similar to the results of identified disposable domains [54]. In our study, however, we do not attempt to achieve high detection accuracy in identifying such a class of domains. Instead, we focus on exploring the efficacy of proposed features on improving the *overall* cache performance by avoiding a large number of once-used records being filled in the cache.

To validate each proposed feature based on its factual caching effect, we *tentatively* derive a threshold to measure the fractions of excluded domains. We then leverage a learning module to train our dataset and build a classifier that combines the proposed features.

3.3.2.1 F1: Length of Query Name

Figure 3.3(a) shows that the distributions of the query name length in which the two classes of domains are clearly distinguishable from each other. The majority of once-used domains apparently have much longer name length than the re-used domains.

To exclude the useless RRs, we start to consider a tentative threshold at 50 bytes of query name length. In WM dataset, a threshold of 50 bytes would exclude 81.10% of once-used domains and 1.61% of re-used domain, resulting in that overall 69.87% of entries are rejected from the cache. By gradually raising the threshold, at the length of 100 bytes we observe that 0.027% of reusable domains are mistakenly dropped while 44.99% of once-used domains (totally 38.77% of RRs) are discarded. Similarly, with the length of 100 bytes, in UD dataset, 0.019% of re-used entries are mistakenly kept out of the cache while 32.58% of once-used domains (totally 25.53% of RRs) are dropped. Such results indicate that rejecting those domains with large names would significantly reduce the waste of cache space but keep the cache hit ratio at the same level.

3.3.2.2 F2: Length of the Longest Subdomain Name

Figure 3.3(b) demonstrates that the large strings are widely adopted in once-used domains. Specifically, with each dataset, we identify 3.03% and 1.51% of re-used entries, as well as 73.40% and 68.05% of once-used entries, include one subdomain with the length of more than 20 bytes, respectively. If we increase the threshold to 30 bytes, the fractions of re-used domains will decline to 0.39% and 0.30% while the fractions of once-used domains remain at 69.86% and 57.22%, respectively. Thus, a length of a subdomain name greater than 30 bytes would strongly indicate that the domain is once-used, with little chance of being a useful entry if cached.

3.3.2.3 F3: Number of Format Fields

Figure 3.3(c) presents the distribution of the total number of format fields. It is easy to identify that a threshold of 10 is capable to distinguish the reusability for each class of domains. Using this threshold, we can exclude 0.59% and 0.79% of re-used entries, and 31.17% and 42.39% of once-used domains, respectively, from each dataset; overall, around 25% and 37% of entries would be discarded from the cache, respectively.

3.3.2.4 F4: Total number of L-FF and S-FF

In order to profile the total number of the *long-format-field* (L-FF) and the *short-format-field* (S-FF), we first empirically determine the specific values of the length to define the L-FF and S-FF. Since most of *TLD*s include one or two fields with two or three characters, we define the S-FF as the fields with the characters less than or equal to three.[1] Also, we investigate the distributions of `F4` by varying the length of L-FF, and observe that a length of 10 is sufficient to demonstrate the distinct statistical properties for this feature.

Figure 3.3(d) shows the distribution of the sum of L-FF ($\geqslant$10 bytes) and S-FF ($\leqslant$3 bytes). With a clear threshold observed at five, we identify that 0.61% of re-used and 70.03% of once-used domains in `WM`'s dataset (overall 60.33% of RRs), and 0.40% of re-used and 74.55% of once-used domains in `UD`'s dataset (overall 63.97% of RRs), would be discarded. As a result, this would exclude the majority of the useless entries but have little negative impact on caching reusable domains.

3.3.3 The Classifier

To validate the efficacy of the combination of proposed features, we train the datasets with both *decision tree* and *random forest* models [117], by using the `rpart` and `randomForest` packages in R, respectively. Note that we leveraged the *class-weights* (i.e., the `parms`

[1] A more accurate approach may need to exclude the TLDs (e.g., `.com` and `.co.uk`) and SLDs (second-level domains, e.g., `msn` and `cnn`) since they may be regarded as the S-FF. However, we observe that checking the entire domain names has already produced effective results, and thus we decide to use a simple way to avoid introducing additional steps to identify the SLDs.

parameter in `rpart` and the `classwt` parameter in `randomForest`) to handle unbalanced class size in our datasets.

Ground truth. Since it is (almost) impossible to have a ground truth that identifies the "disposable domains", we label the once-appeared domains extracted from datasets as disposable domains. As such, our labels correspond to those assigned by an oracle with perfect knowledge. Although the domain unpopularity may cause mis-labeling (i.e., some unpopular domains may be mis-labeled as disposable), our labels are the acceptable approximation to the ground truth in practice, especially given more than thousands of users from each campus network. Moreover, mis-labelling a rarely re-used domain as disposable would have marginal impact on practical caching performance, being it likely to be evicted before reappearing.

Evaluation of the classifiers. Each dataset is divided into mutually exclusive training and testing partitions, where 66% of the dataset is used for training and the rest is used for testing. With the random forest model, we observe that the benefit cannot be achieved with the number of constructed trees higher than five. Table 3.2 lists the percentage of incorrectly classified instances using the combination of all features.[2] Note that we aim to improve the caching effectiveness in two folds: (1) effectively reject the useless entries, and (2) minimize the negative impact on the reusable entries. The results in Table 2 demonstrate that we can achieve both goals in the classification processes. They also indicate that, although the simple decision tree tends to overfit the training set, it is able to produce accurate results when applied to the classifier constructed by the combination of proposed features. More specifically, around 85% to 88% of once-used RRs are correctly labeled and expelled from the cache, while only 0.2% to 1% of re-used RRs are incorrectly classified in `WM` and `UD` datasets, respectively.

The unbalanced (but expected and positive) results can be interpreted by the observation that, the re-used entries have more consistent and concentrated distributions of

[2]We explored the different combinations of feature sets and found that using all features in the classification can achieve the minimum error rate.

Table 3.2: Percentage of mis-classified instances

	Decision Tree			**Random Forest** (ntree=5)		
	D	R	O	D	R	O
WM	16.07%	0.26%	13.88%	12.93%	0.19%	11.16%
UD	13.91%	0.98%	14.13%	11.89%	0.34%	12.18%

D with Disposable, R with Re-used, O with Overall

features, while the extracted features from once-used entries exhibit more diffuse distributions.

Figure 3.4 shows the index of variable importance and the primary split values in decision tree training for each dataset, which illustrates that (1) all the features play important roles in the classification (the importance index varies from 21 to 31) and (2) although the primary split values are more aggressive than the thresholds derived from any single feature (§3.2), we can further lower the error rates by using the combination of features.

3.4 Trace-Driven Simulation

In §3.3, we demonstrate that the explicit domain name-based features are useful to infer the reusability of RRs. However, in practice the resolvers behave slightly differently due to the presence of TTLs. The re-used entries may still cause the cache misses as the cached RRs have been expired. Meanwhile, some mis-classified reusable entries may not affect the caching performance since many of them have a lower possibility to be retrieved again *within* the duration of TTL. In this section, we apply the classifier in §3.3 with the combination of proposed features to conduct a trace based simulation[3] to evaluate the effectiveness of proposed policies.

[3]We only perform the simulation with the WM's trace since the actual domains have been anonymized in the UD's trace.

3.4.1 Implementation

We implemented the proposed caching policies in a simulated resolver program modified from `djbdns` [12], in which the decisions employ the classification results from §3.3. Our resolver program follows the standard TTL model, i.e., do not assign a default minimum TTL value. The duration of negative caching is subject to the TTL values of SOA records [38]. Moreover, we do not set the *cleaning-interval* to periodically expel the stale records due to the use of sophisticated memory management in modern resolvers [6]. Only when hitting the cache limit, will some entries be prematurely evicted from the cache (e.g., using LRU replacement policy).

Types of RRs. First we study the caching properties of different types of RRs. In particular, we examine if they need to be given discriminative considerations when caching in resolvers. Table 3.3 lists the breakdown for the types of queries. The SOAs are treated as same as A/AAAAs, i.e., caching such RRs according to the proposed policies in §3.3. Other unspecified types of RRs are not particularly studied because of their small amounts.

Table 3.3: Summary of RR Types

A	**AAAA**	**TXT**	**PTR**	**SRV**	**SOA**	**NS**	**MX**	**other**†
70.80%	15.33%	5.09%	3.96%	3.86%	0.83%	0.08%	0.03%	0.03%

†Other types include: ANY, NAPTR, DS, DNSKEY, CNAME, AFSDB, AFXR

- **TXT records**: We identify that only 0.01% of TXT records have been reused, and indeed observe that the TXTs are being used as an information channel. We observed the similar distributions of proposed features in TXTs and thus use the proposed policies for caching TXTs.

- **Reserve lookup queries (PTRs)**: We do not apply the modifications on PTRs since PTR is rarely being misused and we do not observe the studied features taking effect on PTRs.

- **Service records (SRVs)**: We identify that most SRVs (97.46%) are involved with the local queries that have been removed in our study (§3.3.1). Like PTRs, we also observe that the studied features have no impact on SRVs' caching effectiveness either, and thus we will not apply the polices on SRVs.

- **NS records**: Caching NS records can significantly enhance the efficiency of DNS and reduce the load on name severs [79]. Also, the number of NS records is much smaller than the other types of RRs above. Thus, there is no need to apply the policies on NS records. In fact, no NS records would be excluded on caching if the proposed policies were applied.

3.4.2 Results

We now evaluate the effectiveness of the proposed policies given a fixed cache memory allocation. To simplify the assessment, we define the cache allocation by the number of RRs. We input the RRs to a cache file and then examine the cache hit rate, which is calculated as the ratio between the number of cache hits and the total number of retrieved RRs.

We need to determine how many RRs should be cached to represent a real scenario for the evaluation of our proposed policies. Jung *et al.* [79] identified that the DNS cache hit ratio is between 80% and 87%. Thus, we choose our cache size as the number of cached RRs that can achieve a similar cache hit ratio. To this end, given the moderate size of our dataset and an FIFO replacement policy, i.e., simply remove the oldest entry when the cache runs out of space, we observe that a size of 100,000 entries would have a cache hit rate about 86%. Therefore, we set the cache size to 100,000 RRs in our simulations. Note that this setting is derived from local campus networks and in ISPs' DNS server, a larger cache size is needed because of the larger number of DNS queries.

3.4.2.1 FIFO

We first evaluate the proposed policies with an FIFO caching scheme, which is still widely used in popular DNS resolvers such as `djbdns` [12]. Figure 3.5(a) presents the measured cache hit rates under the FIFO, with and without proposed polices, respectively. We observe that the modified caching policies can improve the cache hit rate by about 8% with the cache size of 100,000 entries.

3.4.2.2 Pseudo-LRU

We then evaluate the proposed policies with a simplified pseudo-LRU that leverages one bit to store the cache status for each entry (i.e., the MRU-bit). When a cache hit occurs for an entry, its cache bit is set to 1. When the cache is full, the oldest entry with the cache bit 0 is evicted. When the cache bits of all cached entries have been set to 1, all the bits are cleaned to 0 except the last one.

Figure 3.5(b) shows measured cache hit rates with the LRU replacement scheme. We observe that the proposed caching modifications improve the cache hit rate by about 7%. Compared with FIFO, LRU without our policies can increase the cache hit rate by about 2%. With the proposed modifications, both FIFO and LRU can increase the cache hit rate to 92%-93%.

3.4.3 Discussion

TTL values. The lower TTL values have been observed in both malicious domains [47] and disposable domains [54]. However, the domain owners are free to set the TTLs and they have been switching to use larger TTL values (most of them have a TTL of 300s [54]), resulting in the larger duration of the useless entries being regarded as valid in the cache. Meanwhile, since modern resolvers have obsoleted the cleaning-interval [6], caching the once-used domains even for a short time still degrades the performance. This is because the cache is filled with such useless entries and no space is left for caching the useful

entries. Thus, TTL may not be a reliable indicator of the caching effectiveness and we do not consider it as a metric to quantify the caching behavior.

Counteraction. One may concern that the domain owners could circumvent our polices by changing the structure of domain names. However, we believe the modifications will not provoke them to seek more sophisticated approaches since those "one-time use" domains have accomplished the communication mission and the developers using such approaches would not care much if their DNS responses are cached.

3.5 Related Work

DNS Caching and TTL characterization. Pang *et al.* [100] presented a comprehensive study on DNS, and [99] observed that a significant fraction of web clients and LDNSes do not honor DNS TTLs. Callahan *et al.* [50] passively monitored DNS traffic within a residential network to profile the modern DNS behaviors and properties. They also observed that web browsers do not adhere to the given TTLs and CDNs tend to shape traffic with shorter TTLs.

Jung *et al.* [79] presented a detailed analysis on DNS traces to evaluate the client-perceived performance and the effectiveness of DNS caching. They [78] then presented an analytic method of modeling the cache hit rate given consistent TTLs. Choungmo-Fofack *et al.* [64] studied the DNS cache hierarchy, in which the TTLs are overridden by the local values.

Cache modifications. Shang *et al.* [111] proposed an approach to improve the DNS caching by letting the ADNS's response piggyback extra resolution results for future queries predicted by site usage and DNS history. Cohen *et al.* [56] studied the proactive renewal policies in LDNSes, where the expired records are reused to answer the queries and are validated with a concurrent query. Similarly, Ballani *et al.* [44] proposed a minor change in caching behavior of DNS resolvers to mitigate the DNS DoS attacks, where the expired records are stored in a separate "stale cache" and reused to answer the queries

unresponded by authoritative servers.

Malicious domain detection. There have been studies to understand and detect the malicious domains. Hao *et al.* [68] examined the features that may indicate malicious purposes of a domain during its registration. Antonakakis *et al.* [40, 41] and Yadav *et al.* [106] proposed methods to detect dynamically generated malicious domains in DNS traffic. Bilge *et al.* [47] built a passive analysis system extracting 15 features to detect malicious domains. Two of the features, similar to our work, are domain name-based: (1) the percentage of numerical characters and (2) the ratio of the length of LMS (longest-meaningful substring) to the total length of a second-level domain. The most salient difference is that our work focuses on the caching behaviors, regardless of a domain being malicious or benign. Also, the features we used are more simple and straightforward (e.g., the second feature above requires checking the English dictionary).

3.6 Summary

We presented an empirical study on the domain name-based features of DNS queries and exploited these features for improving DNS cache performance. The identified features, including the length of a query name, the length of the longest-subdomain, and the number of subdomains or format fields, are explicitly available from a domain name itself, without involving the deep inspections. Whereas the features do not indicate malicious purposes, the majority of the domains with such properties are indeed associated with the one-time-use pattern and would be highly unlikely to be reused in DNS cache. Our analysis and simulation demonstrate that proactively rejecting such domains from cache can improve the *overall* effectiveness of DNS caching. Finally, we make one of our traces used in this chapter publicly available [8], with proper anonymization but being able to perform the training and simulation.

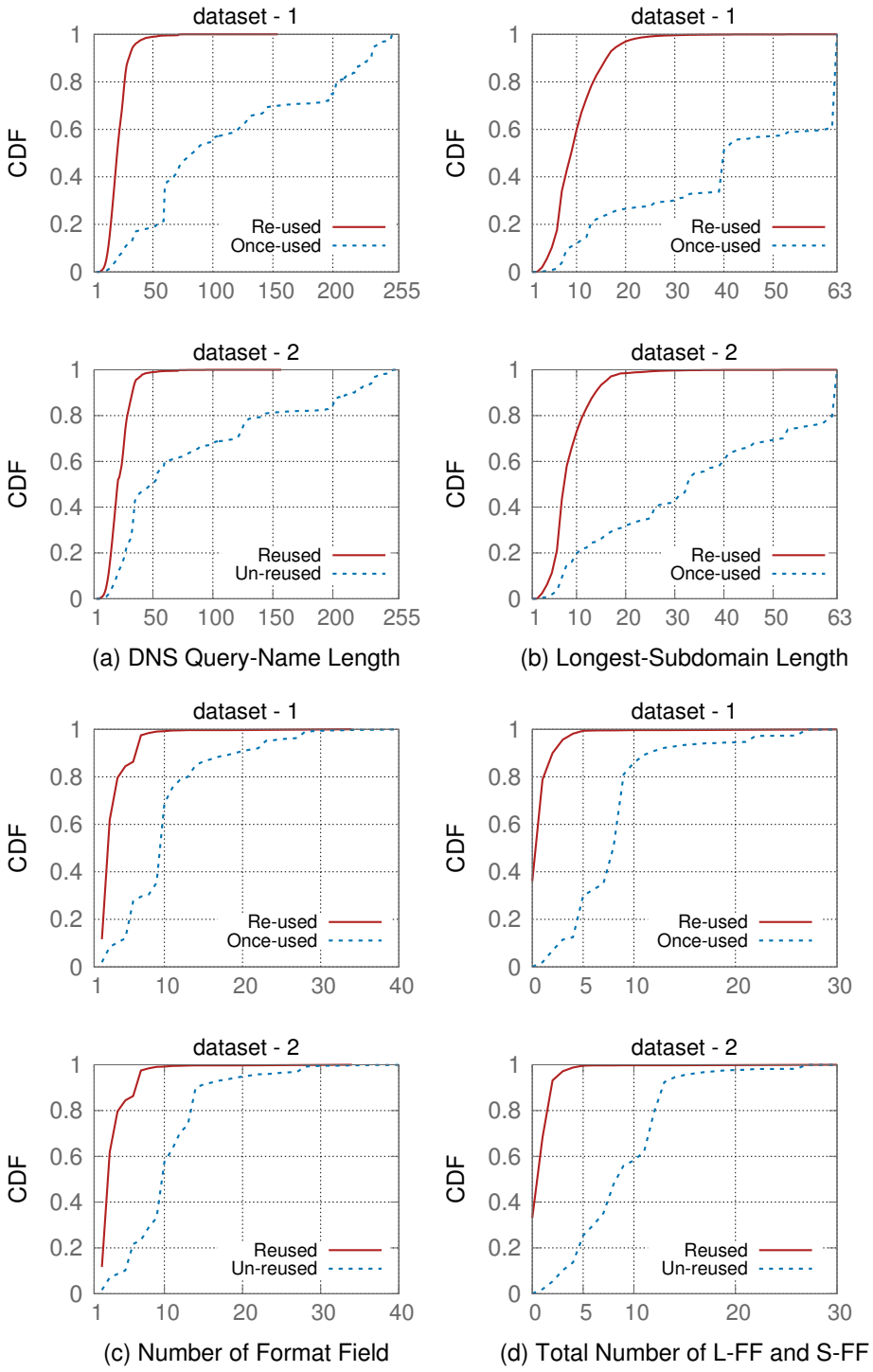

Figure 3.3: Distribution of domain name-based features for `Reused` and `Un-reused` domains

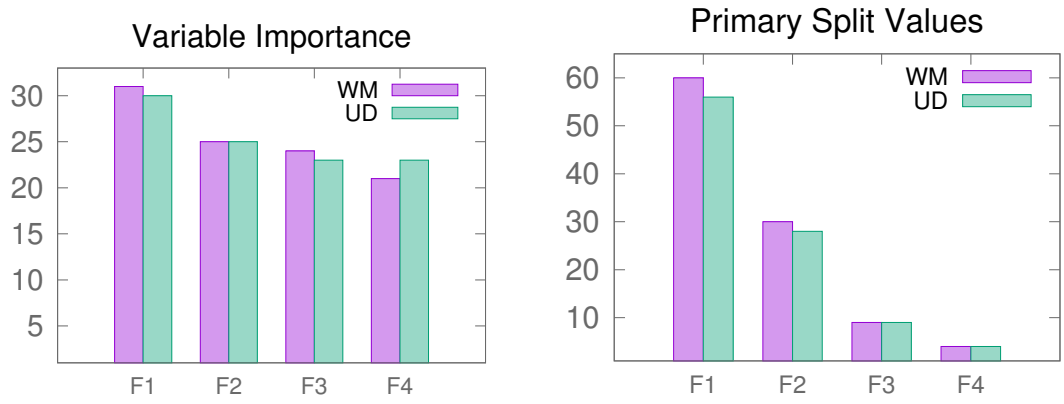

Figure 3.4: Training Results (with Decision Tree).

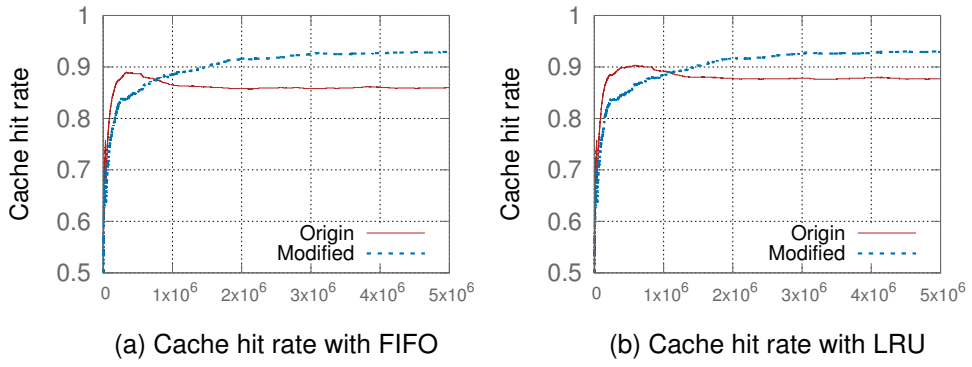

Figure 3.5: Distribution of cache hit rates. The x-axis represents the number of entries read into the resolver.

Chapter 4

End Users Get Maneuvered: Empirical Study of Redirection Hijacking in Content Delivery Networks

Content Delivery Networks (CDNs) place a large number of edge servers (*a.k.a. surrogates*) at geographically distributed edge networks and deliver a large fraction of Internet content to end users by providing content caching and proximal access. The majority of mapping components in today's CDNs rely on the Domain Name System (DNS) to determine the redirection decision for a client's request, i.e., assigning a nearby edge server for the client. However, such a DNS-based mapping system requires DNS records to be dynamically generated in real-time, which restrains those dynamic records from authenticating their mapping DNS records by using DNSSEC signatures since DNSSEC was originally designed for static records. Furthermore, even considering DNSSEC enabled, DNSSEC itself is vulnerable to replay attacks.

In this study, we conduct a large-scale empirical study to investigate the security implications in the DNS-based mapping system of CDNs, which can be exploited by

adversaries to hijack the operation of request redirection in a very stealthy manner. Our work makes following contributions:

- **Illustration of Redirection Hijacking Attacks in CDNs:** While DNSSEC is not designed to immune, the security posture introduced by CDN's dynamic mapping remains understudied. As DNSSEC is vulnerable to a replay attack, we illustrate that an adversary can utilize a *legitimate* mapping record (i.e., a *replayed* message) to override CDN's server selection and redirect a certain group of users to an edge server chosen by the adversary. Worrisomely, even the newly adopted Elliptic Curve Digital Signature Algorithm (ECDSA) that is capable of providing real-time DNSSEC signatures is ineffective to detect and prevent such attacks.

- **Characterization of Operational Practices of CDN's Request Routing:** To assess the magnitude of this vulnerability, we characterize the content delivery operations of popular CDN vendors and perform the threat analysis to elaborate on the ineffectiveness of DNSSEC against redirection hijacking via detailed case studies. We find that 16 out of 20 popular CDN providers suffer from various degree of security threats posed by redirection hijacking.

- **Measurement of Practical Impacts of Redirection Hijacking:** We quantitatively measure the practical impacts caused by redirection hijacking, such as performance degradation and cached content popularity. Moreover, we examine more serious threats, by which adversaries could exploit redirection hijacking to direct end-users to unresponsive edge servers, resulting in the nullification of CDN's benefits (e.g., load balance and DoS mitigation) and the violation of CDN's service commitments.

- **Challenges and Practical Considerations of Countermeasures:** Finally, we present the challenges of addressing this redirection hijacking from different per-

spectives, and elaborate on corresponding countermeasures in practice and their limitations.

The remainder of this chapter is organized as follows. In Section 4.1, we review the background of DNS security and CDN operations. In Section 4.2, we present the threat model and the redirection-hijacking attack. In Section 4.3, we characterize CDN's operations and perform a large-scale threat analysis, illustrating that DNSSEC is not an effective solution. We then discuss the impact of current practice and potential countermeasures in Section 4.4. We survey related work in Section 4.5, and finally we conclude the chapter in Section 4.6.

4.1 Background

4.1.1 DNS Cache Poisoning Attack

DNS provides a vital naming service for users to locate Internet resources by translating domain names to numerical IP addresses, and thus the correctness of DNS resolution is the fundamental anchor for the operation and security of the Internet. Due to its crucial role in accessing Internet services, DNS is an attractive target of adversaries and has been exploited for various malicious purposes. One of the most serious threats to DNS is that adversaries trick a resolver to accept fraudulent DNS records as legitimate responses from authoritative nameservers, known as record injection or cache poisoning attacks [43, 80, 33, 115].

DNS cache is intrinsically vulnerable to record injection, because a recursive resolver cannot ensure whether a received response is from a legitimate authoritative nameserver or a miscreant entity. The general practical approach to mitigating a cache poisoning attack involves the *challenge-response* defenses [77], including transaction-ID (TXID) randomization, source port randomization, or the 0x20 encoding [60], in order to enable a resolver to validate the legitimacy of received responses via the randomized values within the requests.

Although those countermeasures increase the difficulty of injecting fraudulent records, insufficient adoptions and deployment [70, 72, 112] have made many rDNSes still vulnerable to cache poisoning attacks. Large-scale DNS poisoning attacks are still widely observed on the Internet [18, 24, 25]. Furthermore, all those efforts aiming to increase the entropy of DNS queries are only effective against the *off-path* attackers; an adversary, which can monitor network traffic and interpret the transaction packets, is still able to construct a forged DNS response with correct parameters to bypass all the challenge-response defenses and pollute the content of cache, i.e., launch a Man-in-the-Middle (MitM) attack.

4.1.2 DNSSEC

In order to secure the process of DNS resolution, especially against MitM attacks, DNSSEC [42] uses the digital signatures to validate DNS responses. Within DNSSEC, each resource record set (*RRset*) is signed and verified by public key cryptography: a recipient of a signed RRset (i.e., *RRSIG* record) is able to validate the signature via the public key (i.e., *DNSKEY* record) of signer, and the *trust of chain*, starting from "*trust anchor*" at a root zone, ensures that each key is trusted and able to be validated by its parent zone (i.e., *DS* record).

DNSSEC Zone Enumeration. Within DNSSEC, to provide authentication for negative response (i.e., authenticated denial of existence), a Next-SECure (NSEC) record lists and signs a pair of lexicographic consecutive names in the zone, indicating that no names exist between the NSEC's owner name and the "next" name. However, NSEC records expose the existence of names in the zone; this then allows adversaries to enumerate NSEC records and walk through the zone space to learn all the (sub)domains and associated IP addresses, i.e., the zone enumeration attack, resulting in undesired policy violation or more complex attacks [86].

In order to make the zone enumeration more difficult, the alternative NSEC3 record [86] lists the cryptographically hashed names rather than valid (sub)domain names. However,

it is still vulnerable when adversaries apply an dictionary attack by querying non-existent names and guessing real names [9, 67]. Thus, NSEC5 [67] is then proposed to replace the NSEC3's *unkeyed* hash with a new *keyed* hash generated by separate secondary keys.

Another technique to mitigate zone enumeration is "On-line Signing" [125, 114] (i.e., White Lies" [66]). Instead of disclosing real domains or precomputed hashes, on-line signing creates on-demand signature, proving non-existence for a specific name by listing derived predecessor and/or successor. However, this approach has two drawbacks: (1) with traditional RSA algorithm, it introduces significant computational load for authoritative nameservers to generate real-time signatures, resulting in potential denial of service attacks, and (2) the primary private keys have to be distributed among nameservers, increasing the risk of key leakage.

DNSSEC Live Signing by ECDSA. To mitigate zone enumeration and DNSSEC amplification attacks [121], the Elliptic Curve Digital Signature Algorithm (ECDSA) [73] has been employed as an alternative cryptosystem for DNSSEC [122]. Different from traditional RSA-based schemes, ECDSA leverages the Elliptic Curve Cryptography (ECC) to generate signatures with dramatically reduced computational overhead and signature size. More importantly, while validating an ECDSA-based signature is fundamentally slower than validating an RSA-based signature [73, 119], the significantly reduced computational overhead (about 10x faster in signing [10]) allows ECDSA to sign all of the necessary RRSIG records "on-the-fly" [10], i.e., a *live signing* mechanism, providing a potential solution in the context of dynamically generated records at the "edge" of the Internet.

4.1.3 Content Delivery Network

4.1.3.1 DNS-based Mapping System

The mapping system plays a critical role in CDN's request routing for directing each client's request to an appropriate surrogate, which is proximity to a client and has (1) suf-

ficient resource capacity to be responsive and (2) high possibility with requested content cached, leading to an improved user-perceived performance (e.g., low latency). Traditionally, the mapping system uses a client's local recursive DNS resolvers (LDNSes or rDNSes) as the representation of the local area network where a client resides to determine the client's location and assign a neighboring edge server cluster. However, this approach has become inaccurate due to (1) poor location proximity between clients and their LDNSes [110, 92] and (2) increasing usage of public DNS services such as Google Public DNS and OpenDNS. To this end, the EDNS-Client-Subnet (ECS) extension [57] has been proposed to rectify the problem of location discrepancy between clients and their recursive DNS resolvers.

EDNS-Client-Subnet (ECS). Within ECS, the information of the network prefix of a client's IP address (i.e., the local area network from which a query is sent) is included in the option field of a DNS query. Namely, this extension enables the DNS-based mapping system to use the direct knowledge of a client's location, instead of using its LDNS. A recent study by Chen *et al.* [52] showed that Akamai's end-user mapping[1] rolled out by ECS had been providing significant performance benefits for the clients behind public DNS services.

Load Balancing. The load balancing module of a DNS-based CDN such as Akamai typically selects proper surrogates by a two-level assignment [52, 91]: global load balancing and local load balancing. The global load balancing relies on network measurements to select a server cluster, typically geographically close to a client's network. Then, the local load balancing assigns the individual server(s) from the chosen cluster, leveraging the combined information such as responsiveness and capacity.

[1]In [52], the "end-user mapping" is used to dedicatedly describe ECS-based mapping (compared to the NS-based mapping which uses LDNSes). To be clarified, in this chapter we use "DNS-based mapping" to include both ECS- based and NS-based mapping. In most cases, unless specified, we do not differentiate the "DNS-based mapping" and the "end-user mapping" since they have identical implication in the context of dynamic mapping.

4.1.3.2 Anycast Routing

The deployment of the DNS-based dynamic mapping requires extra infrastructure and operational supports. Therefore, some new CDN providers then enable their CDN platforms by leveraging anycast routing, which announces the same IP address(es) from multiple distributed endpoints. BGP routing protocol selects a shortest Autonomous System (AS) path to reach each advertised IP address block, and thus the end users located in different areas will be directed to different topographically-closest locations via BGP routing.

Since anycast-based CDNs rely on the Internet routing protocols for request redirection, conceptually they are immune to redirection hijacking attacks. However, we observe that in practice some anycast CDNs are also leveraging DNS-based mapping to improve accuracy and performance, making themselves vulnerable to request routing manipulation.

CDNs may leverage anycast in different strategies: anycasting nameservers, or anycasting web servers, or both. Note that our study only involves the way that a CDN directs users to web servers. Specifically, anycasting nameservers means that clients will connect the nameservers via their anycast addresses, but does not affect the process of end-user redirection. In particular, if a CDN utilizes anycast DNS but still uses DNS-based redirection, it will also be vulnerable to redirection hijacking attacks.

4.2 Threat Model

4.2.1 Attacker Model

The key feature of a redirection hijacking attack is that an adversary can inject *crafted* but *legitimate* records to a recursive DNS resolver to manipulate the dynamic mapping inside CDNs. We assume that the adversary is capable of bypassing the challenge-response mechanism and injecting records into DNS caches, regardless if DNSSEC is used since DNSSEC itself is vulnerable to replay attack. Specifically, the off-path adversaries can guess the authentication parameters (i.e., source port number and TXIDs) effectively by

applying different techniques (e.g., fragmentation attacks [71, 112] or socket overloading [72]) against the insufficient randomization or vulnerable implementations [70], to launch the record injections remotely and affect a large number of recursive DNS resolvers. It is an ideal scenario that adversaries is on-path and can easily bypass the countermeasures of randomization by sniffing the network packets and observing those parameters. However, such an on-path condition is not required. We assume that adversaries are commonly off-path and can launch a redirection hijacking attack in a highly remote and distributed manner.

Note that recent work [82] has demonstrated that more than 92% of current DNS platforms on the Internet are still vulnerable to record injection; even the popular public DNS platforms are vulnerable to the *indirect injection*, in which a poisonous record is injected in advance and becomes effective after some other records expires. Within CDNs, we assume that adversaries do not have to harvest the surrogate servers [118, 49] or profile CDN's mapping algorithm; they only need to use several selective mapping records to override the CDN's server selection.

4.2.2 Redirection Hijacking Attack

In comparison to the normal operations of a DNS-based mapping system in CDN, Figure 4.1 illustrates how a redirection hijacking attack works: an adversary exploits the dynamic end-user mapping to manipulate an end-user's access to edge networks. Normally, the Content Provider delegates its name resolution to the CDN vendor's mapping system, typically via either CNAME redirection as shown in Figure 4.1 or directly hosting the NS records in CDNs. When a client's request for a content object is redirected into a CDN's nameserver, the mapping component examines the incoming queries (e.g., the client's IP prefix in ECS), performs real-time topological mapping based on network measurements, and returns an optimized assignment that directs the client to a close, responsive edge server [52].

Since the dynamic mapping between end-users and edge servers makes it impractical to

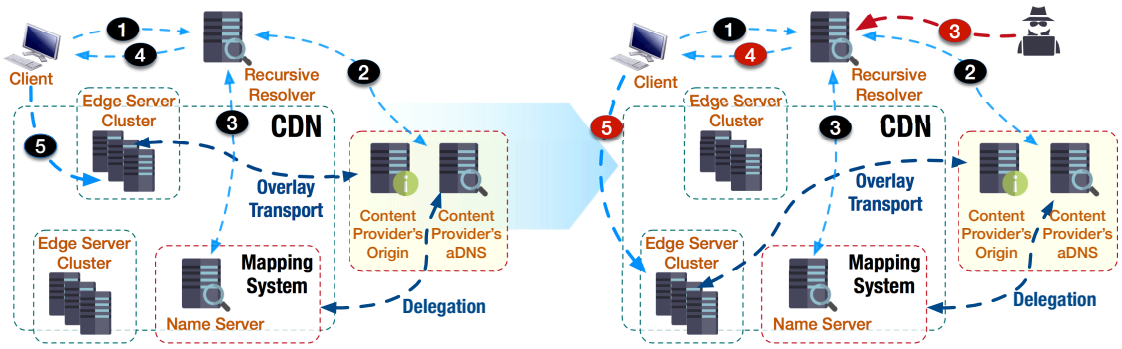

Figure 4.1: Illustration of a Redirection Hijacking Attack

pre-sign a mapping record with traditional RSA-based DNSSEC, we assume that ECDSA is used as the alternative solution to provide on-demand signatures for those dynamically generated records in CDNs. However, even the mapping records with ECDSA signatures are still vulnerable to message replay attacks. This is because (1) in operational practices, the validity period of a DNSSEC signature (including ECDSA) should be long enough[2] to enable easy administration and avoid query load peaks (see §4.4.1 in RFC 6781 [83]) and (2) the validation of DNSSEC signature cannot detect if a message is forwarded or replayed to a different recipient by a third party. An adversary can simply fetch a legitimately signed mapping record that was used or is being used for a different client's network and inject it into the resolver's cache. Because the injected record, which is generated by a legitimate authoritative nameserver but for a different group of clients, carries a valid signature, the resolver will accept it for caching after a successful signature validation. Once the injected record is accepted, the client requests will be redirected to a non-optimal edge server chosen by the adversary, typically heavily loaded and geographically distant from the clients, or even an unresponsive edge server to interrupt the client accesses to the service hosted by CDNs.

We further note that such an attack can be successful even under the environments with

[2]Cloudflare's ECDSA-based signatures are with the validity period of two days. The expiration time of traditional RSA-based DNSSEC signature in practice is normally set to one month [83].

strong security settings. Due to the nature of replay attacks in the redirection hijacking, neither the client end nor resolver signature validations can detect the manipulation.

4.3 Attack Assessment

To assess the magnitude of redirection hijacking attacks in CDNs, we present the characterization of CDN's request routing and conduct a detailed threat analysis to demonstrate the vulnerability of DNS-based CDNs to the malicious manipulation of end-user redirection even with DNSSEC protection. Then, we investigate the practical impacts, such as performance degradation and caching popularity, and explore the more serious threats posed by the redirection hijacking, which nullify CDN's load balancing and DoS protection.

4.3.1 Methodology

In order to identify the CDN platforms that are vulnerable to redirection hijacking, we measure the popular commercial CDNs across the Internet to characterize their configurations and operations. To do so, we set up a group of geographically distributed vantage points on the machines in different Amazon EC2 regions (us-east-1, us-west-2, ap-northeast-1, ap-southeast-2, ap-south-1, eu-central-1, eu-west-1, and sa-east-1, as shown in Figure 4.2) to retrieve the DNS resolution results for customer websites hosted in each CDN provider. Then, we examine the strategies of request routing and analyze the practical impacts, including the more serious threats than performance degradation.

More specifically, we empirically investigate the patterns of content delivery for CDN vendors by taking the steps as follows:

- First, we simply search the official blog articles, technical documents, and announcements published by each CDN vendor, as well as the external technical blogs,[3], to

[3]For example `https://www.cdnplanet.com/blog/which-cdns-support-edns-client-subnet/` and `https://www.cdnoverview.com`.

Figure 4.2: Vantage Points for Resolution

learn the details of content delivery mechanisms.

- We then verify our findings by studying DNS configurations and resolution results from distributed vantage points for a list of customers of each CDN provider, which are gathered by available utilities[4] and the customer list/case studies presented in CDN's websites. For example, an identical `A` RRset should be fetched from different locations when global anycast routing is utilized, and diverse `A` RRsets would be observed when DNS-based dynamic mapping is used.

- Finally, we crosscheck the domain/host names and IP addresses obtained from the DNS resolution by retrieving their information from publicly available passive DNS databases[5] to validate the resolution results.

4.3.2 Characterization Overview

Request routing in CDNs mainly consists of two consecutive steps:[6] *domain delegation* and *surrogate selection*. In the domain delegation, the Content Providers (CPs) delegate the domain resolution to CDN vendors. In the surrogate selection, CDNs redirect a client's

[4]For example `https://trends.builtwith.com/cdns` and `https://wappalyzer.com/categories/cdn`.

[5]`https://www.bfk.de/bfk_dnslogger_en` and `https://www.virustotal.com`.

[6]The higher level techniques of request routing [45] such as application-level request routing are only suitable for large file delivery due to extra latency [52], and thus we only consider those techniques when discussing countermeasures (see Section 4.4.4).

request to a proximal edge server. In essence, these two steps determine how CDNs enable their service infrastructures to be located and accessed by end users. Since CDNs have different ways to perform domain delegation and surrogate selection, we characterize CDNs' request routing with respect to their two redirection steps. Table 4.1 summarizes the request routing and DNSSEC provision in popular CDN vendors.

Table 4.1: Characterization of CDNs' Request Routing and DNSSEC Provision. The "●" indicates that adversaries may be able to manipulate the end-user redirection that results in more serious damage. The "○" indicates that the record suffers from limited form of dynamic vulnerability that may not cause serious threats such as service interruption.

CDN	Domain Delegation	Surrogate Selection	DNSSEC	Dynamics	
			A	CNAME	A
Akamai	CNAME Chain	DNS Mapping (ECS)	×		●
Cachefly	CNAME/NS Hosting	Anycast Routing	Feasible		
CDN.net	CNAME	DNS Mapping	×		●
CDN77	CNAME	DNS Mapping (ECS)	×		●
CDNetworks	CNAME	DNS Mapping (ECS)	×		●
CDNlion	CNAME	DNS Mapping	×		●
CDNsun	CNAME	DNS Mapping	×		●
ChinaCache	CNAME/CNAME Chain	DNS Mapping (ECS)	×		●
CloudFlare	CNAME/NS Hosting	Anycast Routing	✓		
CloudFront (Amazon)	CNAME/NS Hosting	DNS Mapping (ECS)	×		●
EdgeCast (Verizon)	CNAME/CNAME Chain	Hybrid Type I	Feasible		○
Fastly	CNAME	Hybrid Type II	×		●
Highwinds	CNAME	Anycast Routing	Feasible		
Incapsula	CNAME	Hybrid Type I	Feasible		○
KeyCDN	CNAME Chain	DNS Mapping (ECS)	×	●	●
LeaseWeb	CNAME	DNS Mapping	×		●
Limelight	CNAME	DNS Mapping	×		●
MaxCDN/NetDNA	CNAME	Anycast Routing	Feasible		
Rackspace	CNAME Chain	DNS Mapping (ECS)	×		●
cedexis (*MultiCDN*)	CNAME Chain	N/A	×	●	

Domain Delegation. The domain delegation is used to forward each client's request from the origin of Content Providers (CPs) to a CDN's platform. The most common domain delegation mechanisms are CNAME redirection and NS hosting.

- **CNAME Redirection**: The CNAME record enables a domain name to be resolved via an alias. By pointing a CP's domain to a domain provisioned by CDN via CNAME, a client's request will subsequently be redirected to a CDN's domain name and resolved by the CDN's nameservers.

- **NS Hosting**: An alternative approach of domain delegation is to designate CDN-provided authoritative nameservers in NS records of a DNS referral response, which is generated by the CP's authoritative nameservers and then is received by clients. Consequently, the DNS resolution of the CP's domain will be fully operated by CDN.

From Table 4.1, we can see that all CDN vendors provide CNAME redirection to enable the CPs to delegate their DNS resolution to CDNs. By contrast, only three CDN vendors also support NS hosting for domain delegation. Given the prevalent use of CNAME in CDNs, however, we note that the integrity of CNAME records has been widely disregarded on the Internet. This is because (1) typically, the first-level front-end CNAME redirection occurs at the CP's authoritative DNS infrastructure, which is mainly out of control of CDNs, (2) the CP's authoritative nameservers lack of motivations to sign CNAME records, due to the dynamic mapping in the following surrogate selection, (3) in some cases, dynamic CNAME mapping exists in CDN's platforms (see Section 4.3.1), and (4) many CDN providers leverage multiple CNAME records (i.e., CNAME chain in Table 4.1) to facilitate their platform management (e.g., enabling customers to employ different services by being mapped to different CNAMEs), which causes that traversing signed CNAME records are significantly expensive for recursively validating DNSSEC signature for each CNAME record. We will discuss the technique of "CNAME Flattening" in Section 4.4.3 to mitigate the security threat of CNAME in CDNs.

Surrogate Selection. The surrogate selection falls into two fundamental approaches: DNS-based and anycast-based. Table 4.1 shows that the DNS-based mapping is still dominant in CDNs and ECS has been widely supported, especially for those vendors operating a large-scale infrastructure, such as Akamai and Amazon. However, more recent vendors

are more likely to build their platforms with anycast routing to leverage its easy and robust deployment. We also observe that some CDN vendors have employed the different hybrid system design by leveraging both DNS-based mapping and anycast routing to improve the performance of their global content deliveries. In the following, we will elaborate on those different patterns for the operations of CDNs' request routing and analyze the security threat of redirection hijacking caused by the dynamic surrogate selection and the ineffectiveness of DNSSEC via case studies.

4.3.3 Threat Analysis

4.3.3.1 DNSSEC (Live Signing) is NOT a Solution: Case Studies

DNSSEC is proposed as a foundational system-wide solution to DNS vulnerabilities, especially for the record injection by MitM attacks. Here we depict detailed case studies to analyze the vulnerability under different CDN deployment patterns. We demonstrate the infeasibility of providing pre-computed DNSSEC signatures in the dynamic context of DNS-based CDNs. As we discussed in Section 4.1.3, the root cause is that the traditional RSA-based signature algorithm cannot achieve on-demand signature in real-time due to its high computational cost.

Subsequently, for the case studies, we also examine the scenarios when all necessary signature operations can be efficiently performed. To do so, we assume that (1) the CNAME records would be secured by adding corresponding signatures, and (2) CDNs are able to generate on-demand DNSSEC signatures to sign the dynamic mapping records efficiently, such as the ECDSA-based implementation that has been used in CloudFlare's platform [10].

Case Study of End-User Mapping: Akamai. Exemplified by Akamai, Figure 4.3 shows a typical resolution chain by CNAME redirection and the end-user mapping system rolled-out by ECS [52]. Specifically, the domain of the content provider is first translated to a domain provisioned by Akamai's CDN via CNAME. Afterwards, the CDN's nameservers

take over the resolution and finally an A record is dynamically generated by the end-user mapping subsystem to assign an edge server with optimized performance such as responsiveness and capacity, based on the location estimation of the end-user's IP address carried in ECS extension.

Due to the diversity of mapping records and more than 216,000 servers within more than 1,500 networks in Akamai's CDN,[7] it is inefficient and impractical to predetermine or predict the server assignment for each customer and provide a pre-computed DNSSEC signature, resulting in the fundamental vulnerability to record injection attacks. An adversary is able to exploit this vulnerability to hijack redirection and mislead end users to a different domain controlled by the adversary. We note that such a threat can be mitigated by employing ECDSA-based signature, as ECDSA is capable of dynamically signing the records. However, given the adoption of ECDSA, the dynamic mapping is still vulnerable to the redirection hijacking attack as mentioned in Section 3.2.

www.dell.com.	3600	IN	CNAME	www1.dell-cidr.akadns.net.
www1.dell-cidr.akadns.net	3600	IN	CNAME	cdn-www.dell.com.edgekey.net.
cdn-www.dell.com.edgekey.net.	21600	IN	CNAME	cdn-www.dell.com.edgekey.net.globalredir.akadns.net.
cdn-www.dell.com.edgekey.net.globalredir.akadns.net.	3600	IN	CNAME	e28.x.akamaiedge.net.
e28.x.akamaiedge.net.	20	IN	A	104.117.80.33

Figure 4.3: An Example of DNS-based End-User Redirection by CNAME (Akamai)

It is worth noting that, including Akamai, some DNS-based CDN vendors also provide DNS-hosting services with anycast routing and optional DNSSEC signature (e.g., Akamai's Fast DNS[8]). However, this type of service aims to protect the DNS infrastructure itself only; if a customer enables the content delivery, dynamic `A` records are still used to direct end users to edge servers and thus cannot be protected by DNSSEC.

Case Study of Anycast: CloudFlare. Anycast announces the same IP address(es) from multiple locations and relies on the BGP routing protocols to perform the front-end

[7] `www.akamai.com/us/en/about/facts-figures.jsp`.
[8] `www.akamai.com/us/en/solutions/products/cloud-security/fast-dns.jsp`.

redirection. Therefore, the content providers leveraging anycast-based CDNs would have identical `A` record(s), which are static, and thus the anycast-based CDNs are able to secure the integrity of RRsets with either ECDSA-based or pre-computed RSA-based signatures. This makes the anycast-based CDNs immune to redirection hijacking.

The examples below show the configurations of CloudFlare with the domain delegation of CNAME and NS hosting, respectively. In both cases, the returned signed `A` records are with the global anycast addresses, and so there is no risk of redirection hijacking. However, we also note that although the Content Provider enables DNSSEC in the CloudFlare's CDN, the integrity of its CNAME record has been disregarded, which still leads to the risk of domain hijacking.

```
$ DNS resolution for domain using NS Hosting
    filippo.io.        NS      beth.ns.cloudflare.com.
    filippo.io.        NS      jim.ns.cloudflare.com.
    filippo.io.        DS      ...
    filippo.io.        RRSIG   DS      [ECDSA signature]

    blog.filippo.io.   A       104.20.145.15
    blog.filippo.io.   A       104.20.144.15
    blog.filippo.io.   RRSIG   A       [ECDSA signature]

$ DNS resolution for domain using CNAME
    www.martindale.com.                    CNAME   www.martindale.com.cdn.cloudflare.net.
    www.martindale.com.cdn.cloudflare.net .  A      104.18.60.26
    www.martindale.com.cdn.cloudflare.net.   A      104.18.61.26
    www.martindale.com.cdn.cloudflare.net.   RRSIG  A      [ECDSA signature]
```

Note that ECDSA provides CloudFlare the solution to sign their records "on-the-fly" at the edge, but its invulnerability to the end-user manipulation is mainly due to anycast routing rather than ECDSA signing.

Case Study of Hybrid Type I – Regional Anycast: Incapsula. Incapsula enables a hybrid strategy for the request routing, where the DNS-based mapping is used to preliminarily determine the geographic area of end users and a *regional anycast* address is used to serve a specific region. That is, using the regional anycast, a world-wide network is divided into different regions (typically 5-7 regions based on the continents), and within each region, identical anycast addresses are advertised and used to direct end users in this

Figure 4.4: Illustration of Redirection Hijacking with Regional Anycast

region to a close PoP (Point-of-Presence).

On one hand, such a type of hybrid strategy mostly leverages the anycast routing and is able to use DNSSEC to secure the certain number of static DNS records within each region. On the other hand, it still introduces the dynamic DNS records as DNS-based mapping is used, and thus it is vulnerable to the redirection hijacking for maneuvering the end-user access. Figure 4.4 illustrates an example of a global network using regional anycast and its susceptibility to redirection-hijacking. Even with the adoption of DNSSEC, similar to the case of DNS-based redirection, an adversary can inject a legitimate anycast record associated with the end users from a different region, directing the victim users to the edge servers that serve the clients in another continent.

Case Study of Hybrid Type II – Separate Anycast/Unicast: Fastly. Instead of adding the ECS support, Fastly addresses the problem of location discrepancy in a different hybrid strategy: (1) in a normal case, the traditional NS-based mapping is utilized to direct end users to close PoPs; (2) anycast addresses are used to answer the queries from public DNS resolvers. Under such a strategy, the end users behind ISPs leveraging centralized DNS infrastructures will still suffer from the problem of location discrepancy. Moreover, those clients that do not use public DNS services are vulnerable to redirection hijacking, as in the case of DNS-based mapping.

Case Study of Dynamic CNAME: KeyCDN. Unlike other DNS-based CDNs, KeyCDN leverages CNAME to map the CP's domain to a close PoP first, and then assign an appropriate edge server within the PoP via `A` records.

```
$ DNS resolution from us-east
    ja.onsen.io.                CNAME   jaonsenio-4ecf.kxcdn.com.
    jaonsenio-4ecf.kxcdn.com.   CNAME   p-usse00.kxcdn.com.
    p-usse00.kxcdn.com.           A     76.164.234.2

$ DNS resolution from us-west
    ja.onsen.io.                CNAME   jaonsenio-4ecf.kxcdn.com.
    jaonsenio-4ecf.kxcdn.com.   CNAME   p-uswd00.kxcdn.com.
    p-uswd00.kxcdn.com.           A     107.182.231.101
```

The dynamic CNAME mapping introduces another potential attack vector for redirection hijacking via CNAME record. Similar to hijacking a dynamic `A` record, an adversary could inject a legitimate CNAME record associated with a remote non-optimal PoP to degrade the user-perceived performance, even under the availability of DNSSEC live signing enabled by ECDSA.

On the other hand, with the usage of ECDSA, the redirection hijacking for dynamic `A` records would not cause significant performance degradation because all valid `A` records are being mapped to the IP addresses within the nearby PoP assigned by CNAME. However, the adversaries can still leverage legitimate records to redirect users to the IP addresses of unresponsive edge servers within PoP to nullify the DoS protection and interrupt the end-user accesses for the victim service.

Case Study of Multiple-CDN Deployment: cedexis. We then investigate the deployment with multiple CDN providers (*a.k.a. CDN Brokers* [95, 96]). A typical deployment pattern of multiple CDNs leverages Global Traffic Management (GTM) as the first-level redirection, where the GTM platform directs end users to a selected appropriate CDN provider:

```
$ DNS resolution from us-east
    www.lequipe.fr.                     CNAME   2-01-273c-0023.cdx.cedexis.net.
    2-01-273c-0023.cdx.cedexis.net.     CNAME   lequipe-fr.lequipe.netdna-cdn.com.
    lequipe-fr.lequipe.netdna-cdn.com.    A     94.31.29.248
```

```
$ DNS resolution from ap-northeast
    www.lequipe.fr.                  CNAME   2-01-273c-0023.cdx.cedexis.net.
    2-01-273c-0023.cdx.cedexis.net.  CNAME   www.lequipe.fr.edgekey.net.
    www.lequipe.fr.edgekey.net.      CNAME   e7130.g.akamaiedge.net.
    e7130.g.akamaiedge.net.          A       104.116.83.6
```

In the example above, the cedexis's GTM platform[9] is responsible for choosing an appropriate CDN vendor according to the location of a client and the real-time performance of CDNs in this area. First, under such a deployment, the diversity of `A` records depends on the strategy of each CDN's request routing. The clients accessing the website via NetDNA would not be vulnerable to redirection hijacking for `A` records, due to the use of global anycast (assuming signed anycast `A` records), but the clients directed by Akamai will suffer the risk of hijacked redirection mappings.

Since the selection of CDN providers is performed via CNAME redirection, it introduces dynamic CNAME mappings. Thus, DNSSEC live signing cannot prevent the redirection-hijacking attacks, in which legitimate records are injected to redirect users to arbitrary non-optimal CDN providers, nullifying performance improvements offered by both GTM and CDN platforms.

Summary. The vulnerability of CDNs to redirection hijacking stems from the dynamic characteristics of DNS records used for the request routing, which gives adversaries a chance to maneuver CDN's user redirection by injecting crafted but legitimate DNS records. We summarize the features of dynamic mapping for CNAME and `A` records in Table 4.1, respectively. The DNS-based CDNs are widely vulnerable to redirection hijacking, but the CDNs using global anycast for the request routing are immune to redirection hijacking, due to the static mapping of DNS records. Specifically, CloudFlare is the only CDN vendor providing DNSSEC signatures for `A` records to its customers, by leveraging its global anycast routing and ECDSA-based DNSSEC implementation. Also, we consider other CDN vendors with anycast routing of being capable of supporting DNSSEC signatures without DNS dynamics, labeled as "Feasible" in Table 4.1.

[9] `https://www.cedexis.com/`.

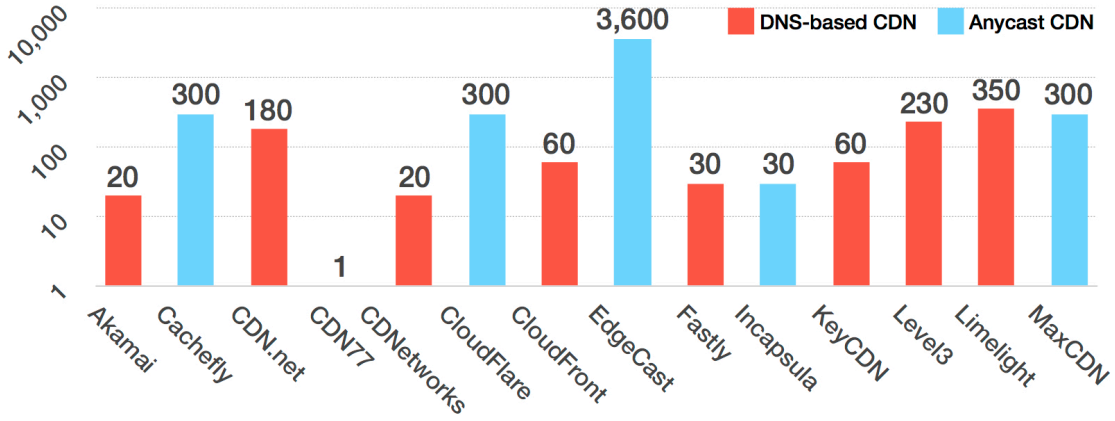

Figure 4.5: TTL

Note that the DNSSEC provision summarized in Table 4.1 involves only the capacity of signing the CDN-issued records for the request routing; the CPs may still be able to sign their records for origin sites, but the request routing would not be protected by their signatures since the mapping records will be provided by CDNs. We argue that this has been a foundational obstacle for the DNSSEC adoption on the Internet, especially for the top websites leveraging CDNs to provide worldwide services.

4.3.3.2 TTL

We list the TTL values of the DNS records for surrogate assignment in Figure 4.5. The DNS-based CDNs use shorter TTL values in their dynamic `A` records for fast traffic redirection and load balancing, typically less than 300 seconds. Most of anycast CDNs have the TTL values of `A` records at 300 seconds, while Edgecast has a larger value at one hour and Incapsula leverages a very short value at 30 seconds.

The length of TTL in a normal DNS record has a significant impact on the possibility of DNS poisoning, because the short TTLs force the recursive resolver to more frequently perform DNS lookups, which grants adversaries more chances (i.e., more frequent "windows of opportunity") to perform the record injections [77]. With DNSSEC enabled, we will craft records based on the legitimate records with valid signatures that are re-used

or replayed. Thus, the prevalent use of short TTL values in normal DNS records indeed increases the possibility of injecting replayed records.

Passive Analysis is NOT effective. Since CDNs typically utilize short TTL values in dynamic mapping records and adversaries usually intend to use larger TTLs in injected records to cause more damage, intuitively, a dynamic record with large TTL value may indicate that it is highly likely to be a crafted mapping. However, popular large-scale passive DNS databases do not enable their sensor servers to capture the TTL information in the traces so that such a manipulation might not be detected via passive DNS databases.

4.3.3.3 Performance Impact

We analyze the performance impact caused by redirection hijacking, in which adversaries inject the records to deliberately direct end users to a geographically distant non-optimal site.

Performance matters. User experience is extremely important to the business of CPs, especially eCommerce sites [52, 7]. Thus, the performance benefits provided by CDNs become critical to the CPs. A prior work [37] observes that even little difference in CDN's performance could cause significant financial gain/loss.

Performance metrics. Similar to the study [52], we measure the following metrics to characterize the potential performance impact when an end user is diverted from optimal edge servers by redirection hijacking.

- Round-Trip-Time (RTT): the RTT measures the propagation delay when the packets traverse the networks, which indicates the quality of the selected network path and is significantly dominated by the distance between two endpoints.
- Time-to-First-Byte (TTFB): the TTFB measures the amount of time between when the first byte of requested content is received and when the client issues the request.

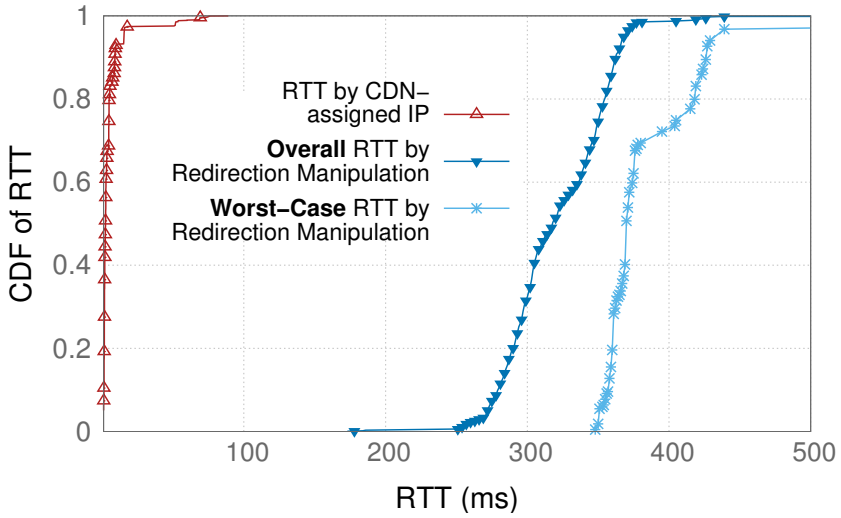

Figure 4.6: CDF for the Round-Trip Time

- Content Download Speed: unlike [52] measuring the page download time, we use the download speed measured by the `curl` utility because the `curl` does not support concurrent connections for embedded contents.

Methodology. We leverage the DNS records obtained via the probes from distributed Amazon regions, as shown in Figure 4.2, and use the same technique for launching a cache penetrating attack presented in [118], where the `curl` utility is used to bypass CDN's server assignment by replacing the normal host header with a (distant) non-optimal IP address in HTTP request. A recent work [53] verifies that such a technique still works for all CDNs in their study. For example, to fetch a content object from an edge server located in Asia as the representation of end users in east coast of Unite States, we issue the following request at a host in the Amazon region of us-east-1:

```
curl -H Host:i.dell.com -O http://104.78.87.26/sites/imagecontent/products
   /...inspiron-15-7000-gaming-pdp-polaris-01.jpg
```

Our experiments are performed based on Akamai's CDN platforms. We manually obtain a list of content objects from popular CDN-hosted sites (dell.com, apple.com, and walmart.com), including static web pages (.html and .css), dynamically generated web pages (embedded search keywords in URLs), images, documents, and medium-sized download

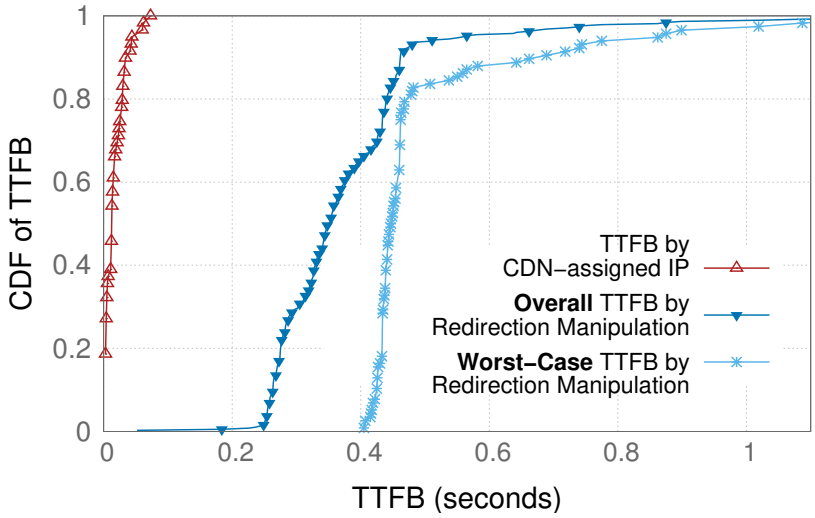

Figure 4.7: CDF for the TTFB

files, with a variety of sizes from 500K to 50M. We download those web contents by using the `curl` utility to evaluate the performance impact experienced by end users under redirection hijacking.

For each metric presented above, we report the measured results associated with the optimal surrogate assignment and the redirected non-optimal surrogates, respectively. In addition, we identify a redirected site with the most significant performance degradation for each vantage point, and plot it as the worst case in Figures 4.6 - 4.8.

Round-Trip-Time. RTT is a purely underlying network latency and the most simple and straightforward performance metric of a network connection and user experience. Figure 4.6 shows that for the optimal assignment of the CDN's mapping system, the RTTs are mostly less than 30ms; but the hijacked redirections typically significantly increase the RTT latency to around 300ms, and in the worst case, the RTTs are increased to around 350 - 450ms.

Time-to-First-Byte. Since TTFB involves both the network latency and the aspects that are not affected by the mapping decisions, such as the construction and compression of a web page, we only include the measured results for web pages. Figure 4.7 illustrates

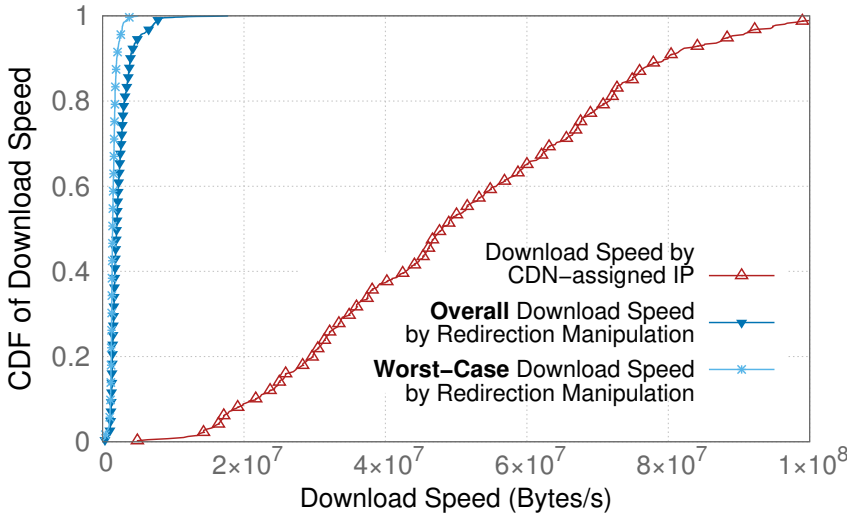

Figure 4.8: CDF for the Download Speed

similar impacts of TTFB in comparison to RTT. Note that our results show lower TTFBs than the results reported in [52], probably due to less dynamics of the web pages we requested.

Download Speed. Figure 4.8 shows the measured speeds for the file downloads. The results from optimal mapping decisions vary, but the cases under redirection hijacking show a significant decrease in their file download performance.

4.3.3.4 Scope of Impact

As discussed before, both the CNAME and `A` records for the CDN's request routing could be exploited by redirection hijacking. We then study whether hijacking a single record can cause collateral damage for other domains. Table 4.2 summarizes the scope of impact for those CDNs vulnerable to redirection hijacking. If CNAME records are unsigned, hijacking a CNAME record itself will just affect the domain associated with this record in all cases, since in these CDNs there is no canonical name being reused among CPs. In other words, there is no shared name appeared in the "left-side" of a CNAME record. However, if CNAME could be signed, only KeyCDN's dynamic CNAME poses the threat of hijacking a single domain. Meanwhile, in some CDNs there could be multiple (sub)domains being

mapped to the same CNAME alias, i.e., a shared name appears in the "left-side" of an `A` record, and thus hijacking such `A` records would have collateral damages for those "co-resident" (sub)domains.

4.3.3.5 Domain Sharding

The *domain sharding* (or *content segregation*) [2] technique is typically used to increase the amount of simultaneous connections by utilizing multiple domains. For example, www.dell.com is directed to e28.x.akamaiedge.net but all embedded images are served via i.dell.com, which is directed to e28.g.akamaiedge.net. Although this technique also distributes the connections to different domains among multiple edge servers, in such a case poisoning a portal domain (i.e., www.dell.com) is sufficient to affect the accessibility of most end users.[10]

4.3.3.6 Impact of CDN Caching

In addition to the issues discussed above, we are aware of that redirection hijacking may also have a subtle impact upon the caching system of CDNs. The caching system is the important building block of a CDN's infrastructure, providing accelerated access for static and popular contents. The cache hit ratio is a critical metric to CDN's performance, since a cache miss may cause extra latency for fetching the requested content from a remote origin server, as well as induce more network traffic and server workload.

The popularity of requested contents on the Internet shows a strong localization. In other words, the redirected end-user groups may be highly likely to have totally different interests on the web contents. Thus, the manipulated redirection would cause the previously cached contents to be rapidly expelled and the limited caches at edge server to be frequently updated, consequently resulting in much degraded performance and user experience. Also, the decreased cache hit ratio will significantly increase the bandwidth costs

[10]Note that the domain sharding would become unnecessary under the adoption of HTTP/2 (SPDY) that supports concurrent requests.

Table 4.2: Impact of a single record hijacking (*CDNs with global anycast that are invulnerable to the redirection hijacking have been excluded.*)

CDN	CNAME		A (signed)	
	Single Domain (unsigned)	Single Domain (signed)	Single Domain	Multiple Domain
Akamai	✓		✓	
CDN.net	✓		✓	
CDN77	✓		✓	
CDNetwork	✓		✓	
CDNlion	✓		✓	
CDNsun	✓		✓	
ChinaCache	✓		✓	✓
CloudFront (Amazon)	✓		✓	
EdgeCast (Verizon)	✓		✓	✓
Fastly	✓		✓	✓
Incapsula	✓		✓	
KeyCDN	✓	✓	✓	✓
LeaseWeb	✓		✓	
Limelight	✓		✓	✓
Rackspace	✓		✓	

of CPs for delivering contents to numerous clients [23]. Finally, the increased back-end connections to origin servers for fetching requested contents will further slow down servers' responsiveness.

4.3.4 More Serious Threats

We further explore the more serious threats of redirection hijacking for maneuvering the end-user's access in CDNs. Technically, CDNs have natural capability to absorb and diffuse attack traffic with the geographically distributed edge networks, and thus become an ideal infrastructure to integrate the enhanced security mechanisms, where the edge

servers can (1) act as reverse proxies to inspect incoming traffic and apply the rules of Web Application Firewalls (WAFs) to filter out malicious traffic and (2) perform the load balancing and DoS protection by diverting users from overwhelmed edge servers via DNS-based dynamic mapping or anycast routing.

Adversaries could exploit redirection hijacking to launch a (parts of) DoS attack by directing the requests from a large number of clients to a single IP address of the victim edge server. The WAFs cannot discard those legitimate traffic from real end users. By selectively injecting the DNS records associated with different popular contents, more clients are connecting to the victim edge server, and then the server has to maintain more back-end connections to different origin servers to fetch the contents. However, the cached contents are quickly being replaced, due to high volume of traffic for massive popular contents. Sooner or later, the victim edge server become overloaded and unresponsive to client requests. More importantly, the load balancing cannot appropriately distribute the traffic since the clients are bypassing the mapping system, and subsequently all the clients that are redirected to those overloaded edge servers will not be able to access the contents or services hosted in CDNs anymore.

Furthermore, the adversaries can leverage the system failure or outage to significantly amplify their attacks. For example, we sent the `ping` probes to monitor the liveness of edge servers for two weeks whose IP addresses have been obtained from our experiments for DNS resolution presented in Section 4.1. We found that 4.5% of IP addresses become unresponsive during the tests, around half of which do not come back online by the end of our experiments. With the easy detection for unresponsive edge servers, the adversaries do not have to perform the actual DoS attack and can simply interrupt end users' accessibility by replaying legitimate mapping records associated with those unresponsive edge servers to resolvers.

4.4 Countermeasures

In this section, we discuss the practical factors affecting the vulnerability and the countermeasures for detecting or mitigating the redirection hijacking attacks.

4.4.1 ECS Considerations

The introduction of EDNS-Client-Subnet provides DNS-based CDNs an attractive scheme to improve the accuracy of their mapping systems and the user-perceived performance for clients using public DNS or the resolvers distant from their locations. As mentioned before, the presence or absence of ECS option does not affect the vulnerability we studied in this chapter. The standardized document [57] does not discuss the difficulty of signing the dynamic mapping records. Also, according to the document, the EDNS0 extension does not change the behavior of data authentication, i.e., the ECS data will not be signed by DNSSEC.

On the other hand, ECS indeed provides another attack vector for DNS abuse. For example, the *scope netmask* carried in ECS indicates the specific IP block associated with a reply. An adversary may be able to selectively poison a resolver's cache to only impact a specific IP range [81], via a fraudulent record directing clients to a malicious address. However, such an activity can be detected if the record is signed by DNSSEC (assuming that either ECDSA is used or only a limited number of mapping records exist so that the signatures can be pre-computed). Furthermore, if adversaries exploit redirection hijacking to maneuver the end-user mapping for tussling CDN's performance or interrupting a service, they could arbitrarily designate the ECS data to impact more clients by using a less detailed network prefix.

Countermeasures. As discussed in Section 4.3, the root cause that even the DNSSEC with live-signing is not effective against redirection hijacking, lies in that the resolvers cannot detect a legitimate but replayed mapping that is supposedly used for a different group of clients. Thus, considering the ECS enabled, one potential mitigation is to include

the ECS data in DNSSEC when signing the RRsets. With ECDSA, the records generated by the end-user mapping can be dynamically signed on demand. Although signing a dynamic record by ECDSA still suffers from the mapping manipulation, due to the ECS data being signed, the resolvers can validate the integrity of the end-user mapping. In other words, adversaries cannot craft a valid record to manipulate the end-user mapping anymore. This is because the signed ECS can guarantee that the data field (i.e., IP addresses) is assigned to the specified user group (ECS data), which eliminates the risk of obscuring the mapping between end users and edge servers.

Limitations. ECS is suggested to be enabled only when clear advantages can be seen by resolvers [57], e.g., open DNS resolvers or a centralized DNS infrastructure serving clients from a variety of geographically distributed networks. Meanwhile, in current practice, CDN vendors typically enable ECS by whitelisting the resolvers that explicitly support ECS, and vice versa. Thus, as only limited adoption of ECS can be expected, signing the RRsets with ECS just authenticates the records in the resolvers that enable ECS.

4.4.2 DNSSEC Considerations

The inclusion of ECS extension as additional information when signing a record with DNSSEC provides an effective countermeasure against the record replay in redirection hijacking, but its effectiveness is limited by the deployment of ECS. Inspired by this, we then consider a more general scheme that leverages existing additional data elements in DNSSEC.

Note that adversaries cannot generate a valid signature since they are unable to obtain the private key. Moreover, the replay attack of redirection hijacking can be successful because the validity period of DNSSEC signatures is typically long enough to be reused by adversaries to launch the record injection. However, only using a much shorter validity period is not sufficient since the signature inception and expiration could also be fabricated

by adversaries. Consequently, we consider that one possible mitigation is to secure the validity period by including additional timestamp information when signing a record. Combined with a short validity period in RRSIG (e.g., only slightly longer than the TTL of mapping records), this would significantly increase the difficulty of record injection as the validity period cannot be altered and adversaries only have a short time window to perform the record injection.

Therefore, a straightforward approach is to include the validity period (i.e., signature expiration) when signing a record. However, since the validity period is associated with the RRSIG record rather than the record being signed, it breaks the normal operations of signing a record (but in a harmless manner): the inception and expiration timestamp will be generate first, and then the signature of RRSIG is computed according to both the responded RRset and the validity period associated with the RRSIG record itself. Correspondingly, the resolver's software needs to be modified to include the expiration when computing the message digest. An alternative approach is to define a new extension representing the validity period in the additional section of DNS messages and sign the RRsets, including such extension data.

Note that the mechanisms we discussed here have the similarities to TSIG/SIG(0) [98, 59], which sign complete DNS request/response with timestamps. However, TSIG requires a symmetric key and thus is most commonly used for authorizing dynamic updates and zone transfers. The SIG(0)'s functionality has been fundamentally replaced by DNSSEC. We argue that it may be worth to enhance the operations of DNSSEC to mitigate the threat of replay attacks under the prevalence of dynamic mapping in CDNs.

4.4.3 CNAME Flattening

One of foundational obstacles for CDN vendors to achieve the integrity of redirection records is the prevalent use of CNAME records, especially the dynamic CNAME mapping and chained CNAME records. A possible solution is to hide the CNAME chain from resolvers and leave the CNAME traversing to CDN's authoritative nameservers, i.e., the

CNAME Flattening [11].[11]

The CNAME Flattening implemented by CloudFare was originally designed to enable the CNAME at the root domain while complying RFC's DNS specification [94], which requires that there should be no other record types if the type of a record is CNAME. With CNAME flattening, the CDN's authoritative nameserver acts as a resolver by recursively resolving the CNAME chain and finally constructs an `A` record to substitute the original CNAME record.

We therefore suggest that the CNAME flattening should also be leveraged by CDNs for security purposes. That is, instead of iteratively replying with multiple CNAME records, the CDN's authoritative nameserver takes the full responsibility of CNAME resolution, typically within the CDN's mapping infrastructure, and finally returns an `A` record, which can be signed with DNSSEC (live signing). This significantly reduces the computational overhead of signing CNAME records as well as the cost of multiple times of signature validation.

Note that the CNAME flattening is mainly associated with the records for the redirection operated by CDNs. The first level CNAME delegation occurs at the CP's authoritative nameservers, which may be out of control of CDNs. However, the CPs can easily secure the CNAME redirections by enabling (traditional) DNSSEC signatures at their authoritative nameservers, since those records are typically static mappings for domain delegation. Also, when enabling the CNAME flattening in DNS-based CDNs, the CDN's authoritative nameservers may need to employ ECS when retrieving the mapping result as the representation of client's networks.

Overall, the CNAME flattening provides CDN vendors with a potential solution to secure the CNAME records at an acceptable cost by avoiding iterative signature validation for multiple CNAME records, while retaining the flexibility of using a CNAME chain to facilitate the platform management.

[11] A similar functionality has also been implemented by DNS-hosting providers, such as the ANAME record [15]. Here we focus on the discussion of such a feature provided by CDNs.

4.4.4 Request Re-Mapping

In addition to performing the request routing via DNS or anycast, CDNs also leverage the high-level re-mapping mechanism to remedy the non-optimal server assignment in some cases. For example, when a request for content objects arrives at an edge server assigned by the mapping system, the edge server first performs an RTT measurement to the client. If the RTT is acceptable, the edge server immediately serves the content to the client based on normal content retrieval strategies; otherwise, the edge server requires the mapping system to reassign an optimal server and direct the client to the different server (e.g., via HTTP status code 3xx for redirection). Due to the extra server selection and redirection operations, the re-mapping introduces additional high latency penalty. Moreover, it is worth to note that, with the wide support of ECS, the accuracy of DNS-based mapping has been significantly improved for those clients impacted by the location discrepancy of LDNSes. That is, the clients are rarely being assigned to a non-optimal edge server. Thus, the request re-mapping is typically only suitable for large file transfer, such as video streaming and software distribution [17, 52].

Nevertheless, CDNs can still enable their Real User Measurement (RUM) system to monitor the performance from a large set of clients and aggregate the monitoring results with geographic locality or client-LDNS pairing to recognize the group of clients affected by anomalous redirections. In general, a more fine-grained performance monitoring and more active request re-mapping could be useful to mitigate serious performance degradation in some cases. However, any high-level re-mapping mechanism still suffers from the threat of nullifying load balancing and DoS mitigation when unresponsive edge servers are exploited in redirection hijacking by adversaries, as discussed in Section 4.3.

4.4.5 DNS Encryption and Transport-layer DNS

DNSCurve [14] and DNSCrypt [13] use ECC to encrypt DNS packets. Google Public DNS offers DNS-over-HTTPS [22] interface to enable the DNS resolution over encrypted

HTTPS connections. Subsequently, Connection-Oriented DNS (T-DNS) [129, 75] is proposed to fundamentally address the weakness of DNS connectionless transmissions in security and privacy. Unlike DNS-Crypt and DNSCurve leveraging the ECC, T-DNS is established over the existing Transport-Layer Security (TLS) framework and is carefully implemented to make the latency and resource needs induced by T-DNS manageable. Using TLS, the channels between stub and recursive resolvers, as well as between recursive resolvers and authoritative servers, would be protected from eavesdropping and MitM attacks.

It is clear that the encrypted DNS and transport-layer DNS indeed address most security and privacy issues of DNS, including the vulnerability we presented in the chapter, because adversaries would be unable to know the content of DNS queries. However, due to high performance penalty and expensive cost for deployment, there is only very limited adoption on the current Internet. Moreover, their negative impacts upon the scalability of DNS, especially at the root and top-level domains, remains unclear.

4.5 Related Work

Disrupting CDN's server assignment has been recently proposed to circumvent the Internet censorship [74, 130], where arbitrary edge servers, rather than the optimal servers assigned by the CDN's mapping system, are used to bypass the DNS-based/IP-based censorship and obtain the censored content. The focus of such censorship circumvention is to retrieve the censored content from edge servers with acceptable performance. By contrast, we explore the attack scenarios where an end-user's access would be significantly degraded or interrupted, resulting in potential financial losses for both CDN providers and content providers.

DNS and CDN. The discrepancy of location proximity between end-users and their LDNSes has been observed for more than a decade [110, 92]. Pang *et al.* [101] characterized the responsiveness of DNS-based network controls according to the behaviors of end-

systems and LDNSes. Huang *et al.* [76] proposed a solution called FQDN extension, where the clients obtain a location-aware cluster identifier and add this identifier to hostnames, to tackle the client-LDNS mismatch problem in Global Traffic Management (GTM). In order to improve the efficiency of content delivery, Krishnamurthy *et al.* [84] proposed a method by which the HTTP interactions are piggybacked on DNS responses. Krishnan *et al.* [85] built a system to diagnose the inflated latencies using active measurements to improve the effectiveness of CDN's indirection and user performance. Scott *et al.* [109] built a tool chain for understanding the web deployment and footprints of CDNs by collecting DNS resolution results and probing the IPv4 address space. In addition, Pearce *et al.* [104] developed a tool to measure and study the global DNS manipulation in Internet censorship.

Ager *et al.* [36] compared the local DNS resolvers against public DNS resolvers (Google Public DNS and OpenDNS) to study the responsiveness and diversity of resolvers. Subsequently, Otto *et al.* [97] examined the performance cost when clients use public DNS services to access CDNs. With the emergence of EDNS-Client-Subnet, Streibelt *et al.* [116] and Calder *et al.* [49] leveraged the ECS with specified client prefixes to infer and profile the large-scale service infrastructure on the Internet such as Google. Kintis *et al.* [81] investigated the potential privacy risk of ECS for surveillance, and revealed a cache poisoning threat for highly selective group of clients.

Cache Poisoning and DNSSEC. Schomp *et al.* [108] assessed the vulnerabilities of diverse record injection attacks, particularly Kaminsky's attack and Bailiwick attack. Duan *et al.* [63] proposed a "Hold-On" period before accepting a reply to mitigate the DNS poisoning by allowing a legitimate reply to also arrive. Lian *et al.* [87] measured the practical impact of DNSSEC deployment and found that DNSSEC-signed domains may create collateral damage in the resolutions of valid domains. Shulman *et al.* [113] developed a validation engine to identify vulnerable keys in DNSSEC-signed domains. Van Rijswijk-Deij *et al.* [119, 120] studied the ECDSA deployment in CloudFlare and the .nl

TLD, and examined the computational overhead induced by the validation of ECC-based signatures. Yan *et al.* [128] proposed a revised DNSSEC signature that constructs a hash chain to limit the replay vulnerability windows when the master server has failed. Their study tackles the malicious slave servers and has a different scope than our study.

Recent studies revealed the pervasive mismanagement of DNSSEC. Shulman *et al.* [113] developed a validation engine to identify vulnerable keys in DNSSEC-signed domains. Chung *et al.* [55] performed a longitudinal measurement study into how well DNSSEC's PKI is managed.

Security Issues in CDN. Liang *et al.* [89] studied the practical impact of CDN's HTTPS deployment. Composing HTTPS with CDN introduces the complexity of authentication delegation since CDN cuts the secure communication paths offered by HTTPS. Similarly, Wählisch *et al.* [124] investigated the Resource Public Key Infrastructure (RPKI) deployment on the routing layer and reported that CDNs are the main cause for the insufficiency of RPKI deployment. While the focus of these work is on the vulnerability of CDN's backend, our study explores the issue of the frontend of CDN's service delivery.

Chen *et al.* [53] presented the forwarding-loop attacks, where malicious customers may be capable of creating the forwarding loops inside one CDN or across multiple CDNs to launch potential DoS attacks. The root cause of this threat is that CDNs lack the control over customers' (mis)configurations. Vissers *et al.* [123] studied the "origin-exposing" attacks to identify the address of a service origin and bypass the cloud-based security infrastructure, typically provided by CDNs.

4.6 Summary

In this chapter, we present a new vulnerability of CDNs, redirection hijacking, which stems from the dynamic characteristics of DNS records used for the CDN's request routing. In a redirection hijacking attack, adversaries can easily maneuver the CDN's mappings between

end users and edge servers by injecting crafted but legitimate DNS records. We reveal that DNSSEC is ineffective to address such a hijacking attack, even with the new ECDSA-based signatures that are capable of achieving live signing for dynamically generated DNS records. This is mainly due to the reusability of signed legitimate records, which can be exploited by adversaries to override CDN's surrogate assignment and redirect client requests to inappropriate edge servers. We assess the magnitude of this vulnerability in the wild by characterizing the operations of the request routing for popular CDN vendors and analyzing the threats via multiple case studies. We quantify the practical impacts of redirection hijacking, especially on performance, and present more serious threats that could nullify CDN's load balancing and DoS protection. Finally, we discuss the countermeasures against redirection hijacking in CDNs from different aspects.

Chapter 5

Conclusions and Future Work

In this chapter, we summarize the studies presented in this dissertation and discuss the future work. The naming system is one of the most critical components of the Internet, directing how can the Internet resources be addressed and accessed. It also provides one of the most fundamental trust anchors, and its authenticity and integrity is implicitly assumed by not only the end users but also the important Internet services that rely on the name resolution, such as web, email, even other critical infrastructures like CDNs.

This dissertation focuses on the different aspects of emerging trends of the Domain Name System and how these evolutions impact the Internet infrastructure and services. We shed light on the characteristics of different deployment patterns for modern web services, propose and validate the simple policies to improve the effectiveness of DNS caching against the prevalent misuse of domain names, and explore the understudied fundamental vulnerability caused by the dynamics of DNS mapping in CDNs, which also restrains the adoption of DNSSEC to secure the integrity of those frequently changed DNS records by signatures.

5.1 Thesis Summary and Contributions

In our first study, we performed a large-scale measurement study to profile the deployment of authoritative DNS servers (ADNSes). We designed a heuristic method to determine

the ADNS patterns, including private, third-party, and hybrid deployment, for a large set of top domains. We then characterized the features for different patterns, such as performance, stability, and availability. In addition, we also studied how the cloud-hosting web services deploy their ADNSes. We collected the subdomains in Amazon's cloud platforms, and profiled the usage for different types of authoritative records, such as the private servers inside or outside the cloud, the third-party DNS-hosting services provided by the cloud itself or external providers.

In our second work, we studied a prevalent class of DNS misuse, the "one-time-used" (a.k.a., disposable) domains in DNS queries. Our key insight is that if those "one-time-used" names that are highly unlikely to be accessed in cache could be easily identified and proactively eliminated, the cache performance could be significantly improved. We investigated and proposed the explicitly available features from domain name itself (i.e., 1. length of query name; 2. length of the longest subdomain name; 3. subdomain depth; 4. number of format fields) to identify the disposability of domain names. We employed a machine learning based approach to build a classifier and combine the studied features. We tested our approach with actual datasets captured at two campus networks, and the results demonstrate that those simple features are able to effectively reject the useless entries (more than 85%), while minimizing the negative impact on the reusable entries (less than 1%).

In our third work, we empirically investigated a fundamental vulnerability of CDNs, called redirection hijacking, which stems from the dynamic mapping of DNS records used for CDN's request routing. Due to the dynamic characteristics of DNS mappings (i.e., assigning a close-by edge server for each request), the pre-computed DNSSEC signatures cannot be deployed to secure those records in CDNs. We further illustrated that even considering the newly adopted ECDSA-based signatures that can achieve live-signing for DNSSEC, it is still vulnerable to a form of replay attacks when adversaries deliberately tamper with the resolvers by injecting crafted but legitimate records associated with sub-optimal or non-responsive edge servers, resulting in significant performance degradation

even service interruption, while remaining undetectable by DNSSEC. We assessed the magnitude of this vulnerability in the wild by characterizing the operations of the request routing for popular CDN vendors and analyzing the threats via detailed case studies. We then presented in-depth discussions on the practical countermeasures against this security issue.

5.2 Future Work

For the future work, there are several interesting directions for extending this dissertation research:

- **DNS measurement**: Complementary to our first study, we may further investigate the usage of spam ADNSes, profile the deployment patterns of TLDs, study the impact of (multiple) CDNs and DNSSEC on different ADNS deployment patterns, and examine more details for cloud-hosting ADNSes. Furthermore, we may perform comprehensive characterization on the dynamics of domain names to investigate the abuse of domain names and how the DNS impacts and facilitates the new threats and attacks on the Internet.
- **Anycast Characterization**: Anycast routing has been widely used in DNS and CDN infrastructures. The most common method to characterize anycast is active measurement, e.g., by using the latency detection of speed-of-light violations. Instead of leveraging probing-based measurement to identify anycast address, we intend to use existing BGP datasets to extract anycast addresses, which gives us ability to (1) illustrate the anycast usage in global routing, (2) a longitudinal study since we can leverage the archived data, and (3) study the usage of IPv6 address space.
- **CDN Vulnerabilities**: Another potential problem we aim to explore is the "residual resolution" problem in CDNs, by which a CDN platform providing the Distributed Denial-of-Service (DDoS) protection may expose the origin IP address when

its customers close the service and/or switch to other platforms, resulting in the failure of protection from future DDoS protection providers.

- **Routing Security**: We would also like to extend our studies to the security issues of Internet routing systems. For example, the Flow Specification [93] has been deployed in major ISPs to provide security rules in global BGP routing infrastructures but remains understudied on its interaction with CDNs.

Bibliography

[1] Akamai Enhanced DNS. http://www.akamai.com/html/solutions/enhanced_dns.html.

[2] Akamai, *Inc.* Customized Caching Rules. https://developer.akamai.com/learn/Caching/Customized_Caching_Rules.html.

[3] Alexa Top Sites. http://www.alexa.com/topsites.

[4] Amazon IP Range. http://docs.aws.amazon.com/general/latest/gr/aws-ip-ranges.html.

[5] Amazon Route 53. http://aws.amazon.com/en/route53/.

[6] BIND 9 Configuration Reference. http://ftp.isc.org/isc/bind9/cur/9.10/doc/arm/Bv9ARM.ch06.html#id2586632.

[7] CDN.net. Why low latency CDN is important for eCommerce stores. https://cdn.net/low-latency-cdn-important-ecommerce-stores/.

[8] Cloud-using Subdomain List. *Data Sets for Cloud Measurement Project*, University of Wisconsin - Madison. http://pages.cs.wisc.edu/~keqhe/cloudmeasure_datasets.html.

[9] CloudFlare, *Inc.* DNSSEC Complexities and Considerations. https://www.cloudflare.com/dns/dnssec/dnssec-complexities-and-considerations/.

[10] CloudFlare, *Inc.* ECDSA: The missing piece of DNSSEC. https://www.cloudflare.com/dns/dnssec/ecdsa-and-dnssec/.

[11] CloudFlare, *Inc.* Introducing CNAME Flattening: RFC-Compliant CNAMEs at a Domain's Root. https://blog.cloudflare.com/introducing-cname-flattening-rfc-compliant-cnames-at-a-domains-root/.

[12] D. J. Bernstein. djbdns. http://cr.yp.to/djbdns.html.

[13] DNSCrypt. https://dnscrypt.org/.

[14] DNSCurve. https://dnscurve.org/.

[15] DNSMadeEasy. Breakthrough in DNS Records. https://www.dnsmadeeasy.com/services/anamerecords/.

[16] Dyn solutions. http://dyn.com/managed-dns/.

[17] E. Zhang. Intelligent User Mapping in the Cloud. https://blogs.akamai.com/2013/03/intelligent-user-mapping-in-the-cloud.html.

[18] F. Assolini. Massive DNS poisoning attacks in Brazil. https://securelist.com/blog/incidents/31628/massive-dns-poisoning-attacks-in-brazil-31/.

[19] G. Sisson. DNS Survey: October 2010. http://dns.measurement-factory.com/surveys/201010/.

[20] Global Traffic Manager, *F5 Networks, Inc.* http://www.f5.com/pdf/products/big-ip-global-traffic-manager-ds.pdf.

[21] Google Public DNS. https://developers.google.com/speed/public-dns/.

[22] Google Public DNS. DNS-over-HTTPS. https://developers.google.com/speed/public-dns/docs/dns-over-https.

[23] Imperva, *Inc.* The Essential Guide to CDN: CDN Caching. https://www.incapsula.com/cdn-guide/cdn-caching.html.

[24] J. Kirk. Google's Malaysia site latest to be felled in DNS attacks. http://www.pcworld.com/article/2054120/googles-malaysia-site-latest-to-be-felled-in-dns-attacks.html.

[25] J. Spring and L. Metcalf. Probable Cache Poisoning of Mail Handling Domains. https://insights.sei.cmu.edu/cert/2014/09/-probable-cache-poisoning-of-mail-handling-domains.html.

[26] Measuring the Health of the Domain Name System. In *Report of the 2nd Annual Global Symposium on DNS Security, Stability and Resiliency*, howpublished = "https://www.icann.org/en/system/files/files/dns-ssr-symposium-report-1-03feb10-en.pdf",.

[27] NetScaler Global Server Load Balancing, *Citrix Systems, Inc.* http://support.citrix.com/article/CTX123792.

[28] OpenDNS. https://www.opendns.com/.

[29] P. Gilmore. Serving at the edge: Good for performance, good for mitigating DDoS. https://blogs.akamai.com/2013/04/serving-at-the-edge-good-for-performance-good-for-mitigating-ddos-part-ii.html.

[30] PlanetLab. https://www.planet-lab.org/.

[31] Public Suffix List, *Mozilla*. https://publicsuffix.org/list/.

[32] Publicsuffix. https://pypi.python.org/pypi/publicsuffix/.

[33] S. Friedl. An Illustrated Guide to the Kaminsky DNS Vulnerability. http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html.

[34] State of the Cloud DNS, *CloudHarmony, Inc.* `https://cloudharmony.com/reports/state-of-the-cloud-dns-report`.

[35] Ultradns. `http://www.neustar.biz/services/dns-services`.

[36] Bernhard Ager, Wolfgang Mühlbauer, Georgios Smaragdakis, and Steve Uhlig. Comparing DNS Resolvers in the Wild. In ACM Internet Measurement Conference (IMC), 2010.

[37] S. M. Nazrul Alam and Peter Marbach. Competition and Request Routing Policies in Content Delivery Networks. In *Computing Research Repository (CoRR)*, 2006. `http://arxiv.org/abs/cs/0608082`.

[38] M. Andrews. Negative Caching of DNS Queries (DNS NCACHE). IETF RFC 2308, 1998.

[39] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a Dynamic Reputation System for DNS. In USENIX Security Symposium, 2010.

[40] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon. Detecting Malware Domains in the Upper DNS Hierarchy. In USENIX Security Symposium, 2011.

[41] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-based Malware. In USENIX Security Symposium, 2012.

[42] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. IETF RFC 4033, 2005.

[43] D. Atkins and R. Austein. Threat Analysis of the Domain Name System (DNS). IETF RFC 3833, 2004.

[44] H. Ballani and P. Francis. Mitigating DNS DoS attacks. In ACM Conference on Computer and Communications Security (CCS), 2008.

[45] A. Barbir, B. Cain, R. Nair, and O. Spatscheck. Known Content Network (CN) Request-Routing Mechanisms. IETF RFC 3568, 2003.

[46] A. Berger, N. Weaver, R. Beverly, and L. Campbell. Internet Nameserver IPv4 and IPv6 Address Relationships. In ACM Internet Measurement Conference (IMC), 2013.

[47] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In Network and Distributed System Security Symposium (NDSS), 2011.

[48] N. Brownlee, K. Claffy, and E. Nemeth. DNS Measurements at a Root Server. In IEEE Global Communications Conference (GLOBECOM), 2001.

[49] Matt Calder, Xun Fan, Zi Hu, Ethan Katz-Bassett, John Heidemann, and Ramesh Govindan. Mapping the Expansion of Google's Serving Infrastructure. In ACM Internet Measurement Conference (IMC), 2013.

[50] T. Callahan, M. Allman, and M. Rabinovich. On Modern DNS Behavior and Properties. In ACM SIGCOMM Computer Communication Review (CCR), 2013.

[51] S. Castro, M. Zhang, W. John, D. Wessels, and K. C. Claffy. Understanding and Preparing for DNS Evolution. In International Workshop on Traffic Monitoring and Analysis (TMA), 2010.

[52] Fangfei Chen, Ramesh K. Sitaraman, and Marcelo Torres. End-User Mapping: Next Generation Request Routing for Content Delivery. In ACM SIGCOMM Conference, 2015.

[53] Jianjun Chen, Jian Jiang, Xiaofeng Zheng, Haixin Duan, Jinjin Liang, Kang Li, Tao Wan, and Vern Paxson. Forwarding-Loop Attacks in Content De-

livery Networks. In Network and Distributed System Security Symposium (NDSS), 2016.

[54] Y. Chen, M. Antonakakis, R. Perdisci, Y. Nadji, D. Dagon, and W. Lee. DNS Noise: Measuring the Pervasiveness of Disposable Domains in Modern DNS Traffic. In IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2014.

[55] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. A Longitudinal, End-to-End View of the DNSSEC Ecosystem. In USENIX Security Symposium, 2017.

[56] E. Cohen and H. Kaplan. Proactive Caching of DNS Records: Addressing a Performance Bottleneck. Journal of Computer Networks, April 2003.

[57] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari. Client Subnet in DNS Queries. IETF RFC 7871, 2016.

[58] C. D. Cranor, E. Gansner, B. Krishnamurthy, and O. Spatscheck. Characterizing Large DNS Traces Using Graphs. In ACM Internet Measurement Conference (IMC), 2001.

[59] D. Eastlake 3rd. DNS Request and Transaction Signatures (SIG(0)s). IETF RFC 2931, 2000.

[60] David Dagon, Manos Antonakakis, Paul Vixie, Tatuya Jinmei, and Wenke Lee. Increased DNS Forgery Resistance Through 0x20-bit Encoding: Security via Leet Queries. In ACM Conference on Computer and Communications Security (CCS), 2008.

[61] P. B. Danzig, K. Obraczka, and A. Kumar. An Analysis of Wide-Area Name Server Traffic: A Study of the Internet Domain Name System. In ACM SIGCOMM Conference, 1992.

[62] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra. Measuring Availability in the Domain Name System. In IEEE International Conference on Computer Communications (INFOCOM), 2010.

[63] Haixin Duan, Nicholas Weaver, Zongxu Zhao, Meng Hu, Jinjin Liang, Jian Jiang, Kang Li, and Vern Paxsonh. Hold-On: Protecting Against On-Path DNS Poisoning. In Securing and Trusting Internet Names (SATIN), 2012.

[64] N. Choungmo Fofack and S. Alouf. Modeling Modern DNS Caches. In International Conference on Performance Evaluation Methodologies and Tools (ValueTools), 2013.

[65] H. Gao, V. Yegneswaran, Y. Chen, P. Porras, S. Ghosh, J. Jiang, and H. Duan. An Empirical Reexamination of Global DNS Behavior. In ACM SIGCOMM Conference, 2013.

[66] R. Gieben and W. Mekking. Authenticated Denial of Existence in the DNS. IETF RFC 7129, 2014.

[67] Sharon Goldberg, Moni Naor, Dimitrios Papadopoulos, Leonid Reyzin, Sachin Vasant, and Asaf Ziv. NSEC5: Provably Preventing DNSSEC Zone Enumeration. In Network and Distributed System Security Symposium (NDSS), 2015.

[68] S. Hao, M. Thomas, V. Paxson, N. Feamster, C. Kreibich, C. Grier, and S. Hollenbeck. Understanding the Domain Registration Behavior of Spammers. In ACM Internet Measurement Conference (IMC), 2013.

[69] K. He, A. Fisher, L. Wang, A. Gember, A. Akella, and T. Ristenpart. Next Stop, the Cloud: Understanding Modern Web Service Deployment in EC2 and Azure. In ACM Internet Measurement Conference (IMC), 2013.

[70] Amir Herzberg and Haya Shulman. Security of Patched DNS. In European Symposium on Research in Computer Security (ESORICS), 2012.

[71] Amir Herzberg and Haya Shulman. Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all.org. In IEEE Conference on Communications and Network Security (CNS), 2013.

[72] Amir Herzberg and Haya Shulman. Socket Overloading for Fun and Cache-poisoning. In Annual Computer Security Applications Conference (ACSAC), 2013.

[73] P. Hoffman and W.C.A. Wijngaards. Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC. IETF RFC 6605, 2012.

[74] John Holowczak and Amir Houmansadr. CacheBrowser: Bypassing Chinese Censorship Without Proxies Using Cached Content. In ACM Conference on Computer and Communications Security (CCS), 2015.

[75] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. Specification for DNS over Transport Layer Security (TLS). IETF RFC 7858, 2016.

[76] Cheng Huang, Ivan Batanov, and Jin Li. A Practical Solution to the Client-LDNS Mismatch Problem. ACM SIGCOMM Computer Communication Review (CCR), 42(2), March 2012.

[77] A. Hubert and R. van Mook. Measures for Making DNS More Resilient against Forged Answers. IETF RFC 5452, 2009.

[78] J. Jung, A. W. Berger, and H. Balakrishnan. Modeling TTL-based Internet Caches. In IEEE International Conference on Computer Communications (INFOCOM), 2003.

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

[80] Dan Kaminsky. It's The End Of The Cache As We Know It. BlackHat, 2008.

[81] Panagiotis Kintis, Yacin Nadji, David Dagon, Michael Farrell, and Manos Antonakakis. Extended Abstract: Understanding the Privacy Implications of ECS. In Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA), 2016.

[82] Amit Klein, Haya Shulman, and Michael Waidner. Internet-Wide Study of DNS Cache Injections. In IEEE International Conference on Computer Communications (INFOCOM), 2017.

[83] O. Kolkman, W. Mekking, and R. Gieben. DNSSEC Operational Practices, Version 2. IETF RFC 6781, 2012.

[84] Balachander Krishnamurthy, Er Krishnamurthy, Richard Liston, and Michael Rabinovich. DEW: DNS-Enhanced Web for Faster Content Delivery. In World Wide Web Conference (WWW), 2003.

[85] Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. Moving Beyond End-to-end Path Information to Optimize CDN Performance. In ACM Internet Measurement Conference (IMC), 2009.

[86] B. Laurie, G. Sisson, R. Arends, and D. Blacka. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. IETF RFC 5155, 2008.

[87] Wilson Lian, Eric Rescorla, Hovav Shacham, and Stefan Savage. Measuring the Practical Impact of DNSSEC Deployment. In USENIX Security Symposium, 2013.

[88] J. Liang, J. Jiang, H. Duan, K. Li, and J. Wu. Measuring Query Latency of Top Level DNS Servers. In Passive and Active Measurement Conference (PAM), 2013.

[89] Jinjin Liang, Jian Jiang, Haixin Duan, Kang Li, Tao Wan, and Jianping Wu. When HTTPS Meets CDN: A Case of Authentication in Delegated Service. In IEEE Symposium on Security and Privacy (S&P), 2015.

[90] R. Liston, S. Srinivasan, and E. Zegura. Diversity in DNS Performance Measures. In ACM Internet Measurement Conference (IMC), 2002.

[91] Bruce M. Maggs and Ramesh K. Sitaraman. Algorithmic Nuggets in Content Delivery. ACM SIGCOMM Computer Communication Review, 45(3), 2015.

[92] Zhuoqing Morley Mao, Charles D. Cranor, Fred Douglis, Michael Rabinovich, Oliver Spatscheck, and Jia Wang. A Precise and Efficient Evaluation of the Proximity between Web Clients and their Local DNS Servers. In USENIX Annual Technical Conference (ATC), 2002.

[93] P. Marques, N. Sheth, R. Raszuk, B. Greene, J. Mauch, and D. McPherson. Dissemination of Flow Specification Rules. IETF RFC 5575, 2009.

[94] P. Mockapetris. Domain Names - Implementation and Specification. IETF RFC 1035, 1987.

[95] Matthew K. Mukerjee, Ilker Nadi Bozkurt, Bruce Maggs, Srinivasan Seshan, and Hui Zhang. The Impact of Brokers on the Future of Content Delivery. In Fifteenth ACM Workshop on Hot Topics in Networks (HotNets), 2016.

[96] Matthew K. Mukerjee, Ilker Nadi Bozkurt, Devdeep Ray, Bruce Maggs, Srinivasan Seshan, and Hui Zhang. Redesigning CDN-Broker Interactions for Improved Content Delivery. In

ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2017.

[97] John S. Otto, Mario A. Sánchez, John P. Rula, and Fabián E. Bustamante. Content Delivery and the Natural Evolution of DNS: Remote DNS Trends, Performance Issues and Alternative Solutions. In ACM Internet Measurement Conference (IMC), 2012.

[98] P. Vixie, O. Gudmundsson, D. Eastlake 3rd, and B. Wellington. Secret Key Transaction Authentication for DNS (TSIG). IETF RFC 2845, 2000.

[99] J. Pang, A. Akella, A. Shaikh, B. Krishnamurthy, and S. Seshan. On the Responsiveness of DNS-based Network Control. In ACM Internet Measurement Conference (IMC), 2004.

[100] J. Pang, J. Hendricks, A. Akella, R. D. Prisco, B. Maggs, and S. Seshan. Availability, Usage, and Deployment Characteristics of the Domain Name System. In ACM Internet Measurement Conference (IMC), 2004.

[101] Jeffrey Pang, Aditya Akella, Anees Shaikh, Balachander Krishnamurthy, and Srinivasan Seshan. On the Responsiveness of DNS-based Network Control. In ACM Internet Measurement Conference (IMC), 2004.

[102] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, and L. Zhang. Impact of Configuration Errors on DNS Robustness. In ACM SIGCOMM Conference, 2004.

[103] V. Paxson, M. Christodorescu, M. Javed, J. Rao, R. Sailer, D. L. Schales, M Stoecklin, K. Thomas, W. Venema, and N. Weaver. Practical Comprehensive Bounds on Surreptitious Communication Over DNS. In USENIX Security Symposium, 2013.

[104] Paul Pearce, Ben Jones, Frank Li, Roya Ensafi, Nick Feamster, Nick Weaver, and Vern Paxson. Global Measurement of DNS Manipulation. In USENIX Security Symposium, 2017.

[105] V. Ramasubramanian and E. G. Sirer. Perils of Transitive Trust in the Domain Name System. In ACM Internet Measurement Conference (IMC), 2005.

[106] A.L. Narasimha Reddy S. Yadav, A. K. K. Reddy and S. Ranjan. Detecting Algorithmically Generated Malicious Domain Names. In ACM Internet Measurement Conference (IMC), 2010.

[107] K. Schomp, T. Callahan, M. Rabinovich, and M. Allman. On Measuring the Client-Side DNS Infrastructure. In ACM Internet Measurement Conference (IMC), 2013.

[108] Kyle Schomp, Tom Callahan, Michael Rabinovich, and Mark Allman. Assessing DNS Vulnerability to Record Injection. In Passive and Active Measurement Conference (PAM), 2014.

[109] Will Scott, Thomas Anderson, Tadayoshi Kohno, and Arvind Krishnamurthy. Satellite: Joint Analysis of CDNs and Network-Level Interference. In USENIX Annual Technical Conference (ATC), 2016.

[110] Anees Shaikh, Renu Tewari, and Mukesh Agrawal. On the Effectiveness of DNS-based Server Selection. In IEEE International Conference on Computer Communications (INFOCOM), 2001.

[111] H. Shang and C. E. Wills. Piggybacking Related Domain Names to Improve DNS Performance. Journal of Computer Networks, August 2006.

[112] Haya Shulman and Michael Waidner. Fragmentation Considered Leaking: Port Inference for DNS Poisoning. In International Conference on Applied Cryptography and Network Security (ACNS), 2014.

[113] Haya Shulman and Michael Waidner. One Key to Sign Them All Considered Vulnerable: Evaluation of DNSSEC in the Internet. In USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2017.

[114] G. Sisson and B. Laurie. Derivation of DNS Name Predecessor and Successor. IETF RFC 4471, 2006.

[115] Sooel Son and Vitaly Shmatikov. The Hitchhiker's Guide to DNS Cache Poisoning. In EAI International Conference on Security and Privacy in Communication Networks (SecureComm), 2010.

[116] Florian Streibelt, Jan Böttger, Nikolaos Chatzis, Georgios Smaragdakis, and Anja Feldmann. Exploring EDNS-client-subnet Adopters in Your Free Time. In ACM Internet Measurement Conference (IMC), 2013.

[117] P. Tan, M. Steinbach, and V. Kumar. Introduction to Data Mining. Addison-Wesley, 2006.

[118] Sipat Triukose, Zakaria Al-Qudah, and Michael Rabinovich. Content Delivery Networks: Protection or Threat? In European Symposium on Research in Computer Security (ESORICS), 2009.

[119] Roland van Rijswijk-Deij, Kaspar Hageman, Anna Sperotto, and Aiko Pras. The Performance Impact of Elliptic Curve Cryptography on DNSSEC Validation. IEEE/ACM Transactions on Networking, September 2016.

[120] Roland van Rijswijk-Deij, Mattijs Jonker, and Anna Sperotto. On the Adoption of the Elliptic Curve Digital Signature Algorithm (ECDSA) in DNSSEC. In International Conference on Network and Service Management (CNSM), 2016.

[121] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. DNSSEC and Its Potential for DDoS Attacks: A Comprehensive Measurement Study. In ACM Internet Measurement Conference (IMC), 2014.

[122] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. Making the Case for Elliptic Curves in DNSSEC. ACM SIGCOMM Computer Communication Review, 45(5), October 2015.

[123] Thomas Vissers, Tom Van Goethem, Wouter Joosen, and Nick Nikiforakis. Maneuvering Around Clouds: Bypassing Cloud-based Security Providers. In ACM Conference on Computer and Communications Security (CCS), 2015.

[124] Matthias Wählisch, Robert Schmidt, Thomas C. Schmidt, Olaf Maennel, Steve Uhlig, and Gareth Tyson. RiPKI: The Tragic Story of RPKI Deployment in the Web Ecosystem. In Fourteenth ACM Workshop on Hot Topics in Networks (HotNets-XIV), 2015.

[125] S. Weiler and J. Ihren. Minimally Covering NSEC Records and DNSSEC Online Signing. IETF RFC 4470, 2006.

[126] D. Wessels, M. Fomenkov, N. Brownlee, and K. C. Claffy. Measurements and Laboratory Simulations of the Upper DNS Hierarchy. In Passive and Active Measurement Conference (PAM), 2004.

[127] K. Xu, P. Butler, S. Saha, and D. Yao. DNS for Massive-Scale Command and Control. IEEE Transactions on Dependable and Secure Computing, 2013.

[128] He Yan, Eric Osterweil, Jon Hajdu, Jonas Acres, and Dan Massey. Limiting Replay Vulnerabilities in DNSSEC. In Workshop on Secure Network Protocols (NPSec), 2008.

[129] Liang Zhu, Zi Hu, John Heidemann, Duane Wessels, Allison Mankin, and Nikita Somaiya. Connection-Oriented DNS to Improve Privacy and Security. In IEEE Symposium on Security and Privacy (S&P), 2015.

[130] Hadi Zolfaghari and Amir Houmansadr. Practical Censorship Evasion Leveraging Content Delivery Networks. In ACM Conference on Computer and Communications Security (CCS), 2016.