2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks

Counting in the Dark: DNS Caches Discovery and Enumeration in the Internet

Amit Klein, Haya Shulman and Michael Waidner

Fraunhofer Institute for Secure Information Technology
Darmstadt, Germany

Abstract

Domain Name System (DNS) is a fundamental element of the Internet providing lookup services for end users as well as for a multitude of applications, systems and security mechanisms that depend on DNS, such as anti-spam defences, routing security, firewalls, certificates and more. Caches constitute a critical component of DNS, allowing to improve efficiency and reduce latency and traffic in the Internet. Understanding the behaviour, configurations and topologies of caches in the DNS platforms in the Internet is important for efficiency and security of Internet users and services.

In this work we present methodologies for efficiently discovering and enumerating the *caches* of the DNS resolution platforms in the Internet. We apply our techniques and methodologies for studying caches in popular DNS resolution platforms in the Internet. Our study includes networks of major ISPs, enterprises and professionally managed open DNS resolvers. The results of our Internet measurements shed light on architectures and configurations of the caches in DNS resolution platforms.

I. Introduction

Domain Name System (DNS), [RFC1034, RFC1035], has a key role in the Internet, and its correctness and availability are critical to the security and functionality of Internet clients, services and networks. Initially designed to translate domain names to IP addresses, DNS is increasingly utilised to facilitate a wide range of applications and constitutes an important building block in the design of scalable network architectures. Over the years, the DNS infrastructure has evolved into a complex ecosystem.

Due to the significance of DNS and its increasing complexity, the research and operational communities invest considerable efforts to gain insight into the inner workings of DNS. Multiple studies were done in order to understand the complexity and security of the DNS infrastructure; we review related work and compare to our research in Section VI.

In this work we focus on one of the most important elements of the DNS infrastructure: the caches. The idea is simple: once a record is requested, it is cached and subsequent requests for the same record will be responded from the cache. This allows to reduce traffic in the Internet and improve efficiency and performance of clients and services. Our goal is to understand the topologies and configurations of the caches, in particular, the correlation and relationship between the IP addresses used to communicate with the clients and the DNS nameservers, and the caches used by them.

Understanding the caches is critical for improving the security and performance of the Internet. Our current view of basic Internet components is based on standard documents and initial designs. However, most systems significantly evolved since their conception. Furthermore, typically the networks or Internet operators make different choices when setting up their infrastructure. In order to evaluate or improve security of the basic Internet components a clear understanding thereof is important. Even basic questions, such as a number of caches in a DNS platform, are necessary to evaluate vulnerabilities to cache poisoning attacks. We explain the motivation for studying the caches in Section II and provide examples.

For our study we design methodologies for measuring the caches and apply them to the DNS platforms in the Internet. The challenge that our study faces is the following: during a DNS transaction, there is no *direct* interaction of the clients and nameservers with the caches. Specifically, the caches are 'hidden' behind resolving devices that communicate with the clients and with the nameservers; see details pertaining to DNS resolution platforms in Section I-A. Our methodology uses the standard DNS request/response behaviour - we trigger specially crafted DNS requests for records in our

2158-3927/17 $31.00 © 2017 IEEE
DOI 10.1109/DSN.2017.63

domains. The DNS requests (including the requested records and their type), that arrive from the resolution platforms at our nameservers, are used as a side channel to infer information about the caches hidden behind the IP addresses of the tested DNS platform. Our methods enable us 'to decouple' the caches from the IP addresses and count the number of caches as well as the mapping between the caches and IP addresses.

Our contribution is twofold: *Conceptually*, our study improves the current understanding of the DNS resolution platforms and serves as a building block for further research on DNS performance and security. In particular, we show that the DNS resolution platforms are more complex than the traditional model of DNS whereby a single cache is situated behind a given IP address. *Practically*, our tools and measurements can be used for improving the security, efficiency and consistency of DNS as well as for designing mechanisms that utilise the DNS infrastructure.

A. DNS Resolution Platforms

We consider a general model for DNS resolution platforms, illustrated in Figure 1. Typically a full subnet is allocated for the resolvers (as shown in Figure 1) but that is not mandatory for our analysis. The platform consists of a set (2^{32-x}) of ingress IP addresses which handle DNS queries from the clients, a set of n caches, and a set (2^{32-y}) of egress IP addresses, which communicate with the nameservers if the queries from the clients cannot be satisfied from (one of) the caches. The load balancers apply logic for selection of the caches to sample, and for an egress resolver's IP address (in case a requested record cannot be satisfied from caches).

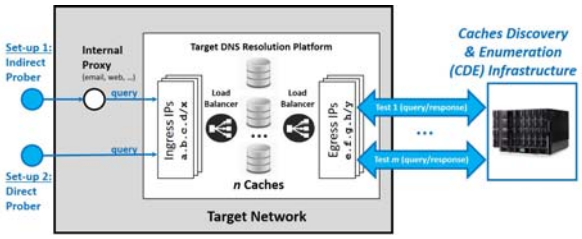

Fig. 1. A generic view on DNS resolution platforms and the relevant actors.

This infrastructure corresponds to complex platforms such as Google Public DNS, and it can also be abstracted to incorporate a very simple version for a DNS resolution platforms with a single IP address which performs both the ingress and egress functionalities and uses a single cache.

Our measurement infrastructure, *Caches Discovery and Enumeration (CDE) Infrastructure* uses direct and indirect probers to communicate with the ingress IP addresses of the DNS resolution platforms, and a set of nameservers to communicate with the egress IP addresses of the DNS resolution platforms.

B. Contributions

We design tools for studying the caches in DNS resolution platforms in the Internet, and report on the results of our large scale evaluation of caches in diverse networks populations. Our analyses provide insights on architecture of caches in DNS platforms.

Our study is performed on a diverse and representative dataset of networks. For our data collection we utilise three different approaches: (1) a distributed ad-network, (2) email servers in popular domains, (3) popular Alexa networks (`www.alexa.com`) that operate open resolvers to reduce traffic to nameservers (see details in Section III). The evaluation on different types of networks offers an accurate and representative view of the DNS resolution platforms in the Internet. Prior studies on DNS resolvers were typically done either on the nameservers' side (by collecting queries that arrive at the nameservers) or by scanning the IPv4 address block for *open* recursive resolvers.

Our tools enable repetitive studies of the caches over periods of time. This allows to perform analyses of adoption of new mechanisms, trends, growth of the DNS resolution platforms and more. We make our tools available for public use[1].

C. Organisation

In Section II we explain motivation for our study and describe examples where the tools and the statistics can be useful. In Section III we present our dataset and data collection study, and report on the insights important for characterising the DNS infrastructure (such as IP to caches mapping). In Section IV we design techniques for caches enumeration and for mapping between IP addresses and caches. In Section V we apply our techniques for a study of DNS resolution platforms in the Internet. In Section VI we compare our study to related work. We conclude this work in Section VII and provide directions for future work.

II. Motivation for Studying Caches

Understanding Internet components is critical for improving their security and for designing mechanisms that use them or depend on them. In this section we explain the importance of studying the caches in DNS platforms.

[1]Due to double blind requirement, the tools are available upon request from the authors.

A. Security

The structure of DNS platforms and the number of caches that they use are important parameters in assessing the security of networks against cache poisoning attacks. Using multiple caches significantly increases the difficulty of cache poisoning. During a cache poisoning attack, spoofed responses, containing incorrect DNS records, are sent to the victim DNS resolver. The resolver accepts only *correct* responses, namely those with correct challenge response authentication parameters (defined in [RFC5452]). However, recent work showed that often the defences can be circumvented, [1], [2], [3], [4], [5], [6], [7]. As a result, an off-path attacker can inject a spoofed response that would be accepted and cached by the resolver. A man-in-the-middle (MitM) attacker does not even need to attempt to guess the challenge response authentication, but can simply copy the values from the request to the response. Typically a cache would already contain the values which the attacker attempts to inject, hence the attacker would need to overwrite the cached records with ones that contain spoofed values, [8]. This requires exchanging a sequence of request/response pairs with the same cache. In a multiple cache scenario the difficulty to launch a successful cache poisoning attack increases significantly. For instance, different caches apply different logic for deciding which records to cache, hence the attacker would need to force his attack to run against a specific cache with the vulnerable software. Furthermore, the spoofed records sent by the attacker will be distributed to multiple caches, hence rendering the attack ineffective - say if an attacker wishes to inject an NS record and then to use it to supply a spoofed A record for a website. In that case, if one of the records 'hits' a different cache, the attack fails. Understanding the structure of DNS platforms and the number of caches that they use enables researchers and network operators to evaluate resilience and security against cache poisoning attacks.

B. Resilience

Tools for studying caches allow to model and analyse the dependability and security of DNS resolution platforms without a priori knowledge of their structure. For instance, how resilient is a DNS platform and the services that depend on it against Distributed Denial of Service (DDoS) attacks. How easy is it to disconnect services from the Internet by attacking DNS? Tools developed in this work help network operators to assess the security and dependability of their networks. For instance, a network operator can identify when some of the caching components fail and are not available, e.g., a DNS platform uses four caches, but our tool measures two, namely two are down. Our tools can also be used for large scale studies of the dependability of the Internet while being non intrusive and requiring no cooperation from the tested networks.

In addition to facilitating further research on resilience of networks and platforms in the Internet, our tools provide means for comparing DNS service operators in order to select one with better resilience and fault tolerance.

C. Tools for Networks Management and Research

In this section we provide a number of examples, in which caches enumeration is critical and useful.

- *Consistency of the caching of DNS records*. Our study provides tools for differentiating between multiple caches and inconsistency in caches with respect to DNS Time-to-Live (TTL). For instance, requesting the same record from a DNS resolution platform twice, within a time interval that is shorter than the TTL of the DNS record, should result in only one DNS request issued by the resolution platform to the nameservers[2]. Current studies interpret multiple requests as inconsistency with TTL. However, it can also be that the DNS resolution platform is using multiple caches. When a DNS platform is configured with multiple caches, not all the caches will have the requested record, hence will issue the query again. This can be mistakenly taken as an indication that the DNS platform does not respect the TTL. Our tools allow researchers and network operators to differentiate between multiple caches and caches with inconsistent TTL.
- *Size of DNS resolution platforms*. Originally designed to translate domain names to IP addresses, DNS is increasingly used by different mechanisms, for instance mechanisms for security, such as DANE, anti-spam mechanisms, routing defences against prefix hijacks. As a result, the DNS resolution platforms have to satisfy the increasing demands for storage and the traffic volume. Our tools enables measuring the sizes of DNS resolution platforms in the Internet.
- *Device discovery*. Device discovery mechanisms find devices that have addresses, since caches are not identified by IP addresses, they would not be detected. Our tools extend current device discovery mechanisms enabling discovery of hidden caches . Our ideas and techniques can also be applied for design of discovery of other hidden caches.
- *Measuring software and new mechanisms*. Distribution of patches and upgrades. Caches on DNS resolution platforms are often running different DNS software. For distribution and integration of patches it is important to know which software the caches are

[2]Some DNS resolution platforms enforce a minimal and a maximal TTL. In those cases, TTL that is smaller than the minimum, or larger than the maximum will be adjusted by the cache.

running. Inferring the software used by DNS caches is also important for the studies measuring deployment of DNS software in the Internet. As we show in related work, currently, studies on DNS resolution platforms measure devices with IP addresses but omit the hidden caches. Similarly our tools enable studies of adoption of new mechanisms for DNS, such as the transport layer EDNS [RFC6891] mechanism.

III. Data Collection

We perform our study of DNS resolution platforms on a dataset of diverse networks, which comprise: (1) open recursive resolvers, (2) resolvers operated by Internet Service Providers (ISPs), and (3) resolvers serving enterprises. Our data collection methodology generates active probes to the DNS resolvers in the target (tested) networks directly in case of networks operating open recursive resolvers and indirectly via email servers or web browsers.

In contrast to prior work on DNS, which typically collect data from open resolvers, our work provides a wide view on different types of networks. We explain the data collection for our dataset using open resolvers, email servers and ad-network in Sections III-A, III-B and III-C respectively.

In Figure 2 we list top ten Internet Service Providers (ISPs) of the DNS platforms in our dataset (each column represents a distinct dataset).

A. Collecting Data with Open Resolvers

Our dataset of networks operating open resolvers contains popular networks, and excludes malicious networks and home networks. We discuss our data collection and show that the networks are *well managed* and often *security aware*.

Our population of networks running open resolvers includes public services, Google Public DNS and OpenDNS as well as 1K networks operating open resolvers among top-10K Alexa networks (taken from `www.alexa.com`).

We obtain the IP addresses of 1K domains containing open resolvers out of top-10K Alexa networks in two steps: (1) we queried the top-10K Alexa domains for nameserver (NS) records, and their corresponding IP addresses (A records); (2) we select the first 1K domains that provide open DNS resolution services, by querying these IP addresses for records in our domain. This resulted in 1739 IP addresses located in 63 countries, and hosted by 1532 Autonomous Systems (ASes). These IP addresses correspond to open resolvers whose main purpose is to reduce traffic to the nameservers, by responding to the clients' queries from the cache. The open resolvers are transparent to the clients, since the IP address of the resolver is used as the A records of the nameservers in the zone files of the corresponding domains, and the nameservers are essentially hidden behind the IP addresses associated with the open resolvers. Specifically, when receiving a DNS request, the value is checked in the cache, and if it is not in the cache, it is relayed to the nameserver. The IP address of the nameserver is hidden and only the resolver communicates with the nameserver directly.

B. Collecting Data with Email Servers

We collected a set of DNS resolvers using email server in the top-1K enterprise networks according to Alexa[3]. We establish an SMTP session to each SMTP email server in the list of enterprise networks, over which we sent an email message to a *non-existing* email-box in the target domain, that the SMTP server is responsible for. Upon receipt of email messages, the SMTP servers trigger DNS requests via the local recursive resolvers in order to locate or to authenticate the originator of the email message. Since the destination is a non-existing recipient, the receiving email server must generate a Delivery Status Notification (DSN, or bounce) message to the originator of the email message informing the sender that the message could not be delivered. The rule to send bounce messages is mandated by [RFC5321], to enable the originator of email messages to detect and fix problems and prevent email messages from silently vanishing.

The email server, sending the bounce message, has to perform some DNS resolution via its local DNS resolver, typically searching for `MX` and `A` records of the target email server, but also possibly via other DNS request types.

In Table I we list the DNS request types' triggered by the resolvers in the 1K domains with emails that we surveyed. We use the queries of the resolvers to initiate our study (described in subsequent sections).

Query type	Fraction
Modern SPF queries (TXT qtype)	69.6%
Obsolete SPF [RFC7208] (SPF qtype)	14.2%
ADSP (w/DKIM)	2%
DKIM	0.3%
DMARC	35.3%
MX/A queries for sending email server	30.4%

TABLE I
DNS QUERIES GENERATED DURING THE SMTP POPULATION DATA COLLECTION.

C. Collecting Data with Web-Browsers

We used an ad-network to collect data from the resolvers used by web clients. The vast majority of

[3]Alexa website also provides ranking of networks according to different categories. We used top-1K networks of enterprises.

the clients attracted through an ad-network were from networks of different ISPs.

For our study, we embedded our script (which is a combination of Javascript and HTML) in an ad network page, and placed it at a static URL. Our script is wrapped in an iframe by the ad network, and iframe is placed on webpages. When downloading the web page, the Javascript causes the browser to navigate to our URLs, which generates DNS requests to our CDE infrastructure Figure 1. We received more than 12K web clients, in which an AJAX call was made to our web server (indicating the page was loaded and functional, i.e., Javascript running). Our test ran as a *pop-under* and needed several minutes to complete. Out of 12K clients, approximately 1:50 of the executions resulted in tests that completed successfully.

IV. Caches Discovery and Enumeration

Understanding and characterising the caches in resolution platforms is prerequisite for: identifying vulnerabilities in caches, such as those exposing to cache poisoning attacks, for hardening caches against attacks, e.g., by designing secure caching policies or cache selection algorithms, for upgrading DNS platforms with the required resources, for extending DNS to support new systems that use the DNS infrastructure and for facilitating research on caches in the Internet.

In this section we present our methodology for characterising caches and then in Section V we apply it for evaluation of caches in the resolution platforms in the Internet. Our study in this section is comprised of two parts: we design techniques for caches discovery and enumeration and we learn the mapping between the caches and the set of ingress and egress IP addresses of a given DNS resolution platform. We use a prober to initiate our study by triggering DNS queries either directly via the ingress IP address of the DNS resolution platform, or indirectly, via email server or web browser (see Section I-A). A direct prober can issue DNS request the ingress IP address of the resolution platform. In contrast, when using the latter (email or web browsers) we do not know the IP address of the ingress resolver and do not communicate with it directly. Communication between the prober and the DNS resolution platform enables us to create a mapping between the prober and the set of caches.

Then, we use the communication between the set of egress IP addresses and our *Caches Discovery and Enumeration* (CDE) infrastructure for discovery of the egress IP addresses used by DNS resolution platform.

We describe methodology and then present methods for counting the caches and characterising the cache to IP mappings. We discuss the challenges in discovering and enumerating caches when a direct access to the target resolution platform is not available, i.e., when the tests are carried out via web browsers or email servers, and show how to overcome this limitation.

A. Setup and Methodology

The setting that we consider consists of a target DNS resolution platform and of our *Caches Discovery and Enumeration* (CDE) infrastructure, see Figure 1. All the communication channels between our CDE infrastructure and the DNS resolution platform are illustrated in blue. The CDE infrastructure owns a domain `cache.example` and uses subdomains, under `cache.example`. It also utilises nameservers, authoritative for `cache.example`, and nameservers authoritative for the subdomains of `cache.example`. The probers initiate the study of DNS resolution platforms by triggering DNS queries. The study is composed of a set of q DNS requests to the (direct or indirect) prober. The ingress DNS resolver is configured to use one or more caches; let n be the number of caches. The queries, arriving at the resolution platform, are assigned to caches by means of a load balancer. The load balancer is situated on the DNS resolution platform (see Figure 1).

Resolution platforms use different cache selection methods for probing caches. Within our study we identified two cache selection methods: traffic dependent (which attempt to evenly distribute the queries' volume to caches) and unpredictable. An example of the former category is *round robin* cache selection, where the next cache is probed each time a new query arrives. A *random* cache selection is a representative of the unpredictable category, where a randomly selected cache is probed next. We also identified more complex cache selection strategies, e.g., those that look not only at the volume of the arriving DNS queries but are also a function of a requested domain in the query or of a source IP in a DNS request. Our measurement indicates that more than 80% of the networks in our dataset support unpredictable cache selection. A comprehensive study of cache selection algorithms is outside the scope of this study and we propose it as one of the interesting followup topics for future work.

During each iteration, i.e., when the ingress resolver receives a DNS query, exactly one cache is selected by the load balancer for sampling. If a DNS record, corresponding to the query, exists in the selected cache (i.e., a *cache hit* event occurs), the query is responded from the cache. Otherwise, if no corresponding value exists in the cache (a *cache miss* event occurs), the query is sent by the egress DNS resolver to the nameserver, authoritative for the domain that was in the query. Our

Open Resolvers		Email Servers		Ad-Network	
Network Operators	**%**	**Network Operators**	**%**	**Network Operators**	**%**
Aruba S.p.A.	9.597	Google Inc.	24.211	Comcast Cable Communications, Inc.	15.02
Google Inc.	6.59	Yandex LLC	10.526	Time Warner Cable Internet LLC	6.103
Korea Telecom	4.095	Amazon.com, Inc.	4.2105	Orange S.A.	5.634
INTERNET CZ, a.s.	3.199	Hangzhou Alibaba Advertising Co.,Ltd.	4.2105	Google Inc.	4.695
tw telecom holdings, inc.	3.135	Internet Initiative Japan Inc.	4.2105	BT Public Internet Service	4.225
LG DACOM Corporation	2.687	Websense Hosted Security Network	4.2105	MCI Communications Services, Inc. Verizon	3.286
Data Communication Business Group	2.175	SAKURA Internet Inc.	3.1579	AT&T Services, Inc.	2.817
Getty Images	1.727	ADVANCEDHOSTERS LIMITED	2.1053	OVH SAS	2.817
CNCGROUP IP network China169 Beijing	1.536	Dadeh Gostar Asr Novin P.J.S. Co.	2.1053	Free SAS	2.347
Level 3 Communications, Inc.	1.536	Limited liability company Mail.Ru	2.1053	Qwest Communications Company, LLC	2.347
OTHER	63.72	OTHER	38.947	OTHER	50.7

Fig. 2. Distribution of Internet Network Operators across the networks in our dataset.

study proceeds by observing and counting the number of queries arriving at our nameservers.

In the following sections we describe techniques, that use this setup, to characterise the caches, including caches enumeration and mapping. We report on Internet measurements of our techniques in Section V.

B. Techniques and Tools

Our study of the DNS resolution platforms is performed on three dataset of networks: ISPs (via web browsers), enterprises (via SMTP), and networks which provide open resolution services. Applying our methodology to the DNS resolution platforms in each population requires facing some challenges, as we next explain. When studying open recursive resolvers (set-up 2 in Figure 1), our prober has a direct access to the ingress IP address, and can control the timing of the queries and the number of times a given query is issued. In contrast, when performing tests via email servers and web clients our prober has only an indirect access to the ingress resolver in the target DNS resolution platform (set-up 1 in Figure 1), and has to bypass the local caches and proxies between the *internal proxy* (set-up 1 in Figure 1) and the ingress IP address of the resolver. The local caches include caches in operating systems, caches in stub resolvers, caches in web browsers and web proxies; for instance, a local cache within the browsers, such as Internet Explorer or the stub DNS resolver's cache within the operating systems, such as Windows8. The intermediate local caches introduce two challenges: (1) each hostname can be queried only once (the subsequent queries for that name are responded from the local cache without reaching the ingress resolver - until its time-to-live (TTL) expires), and (2) during the test we do not have control over the *timing* of the queries. We explain our approaches for bypassing the local caches in Section IV-B2.

Another challenge is related to a direct or indirect access to the egress DNS resolver in the target resolution platform. A direct access to egress IP address assumes that our CDE infrastructure can attract DNS requests for resources within the domain that we own, i.e., such we can observe the queries arriving at our nameservers. However, a direct egress access may not always be available. For instance if it is desirable not to 'leave traces' in the logs of a domain used for the tests (say, if the caches study is performed by an attacker as part of an Advanced Persistent Threat (APT) attack), or if the users and servers in a target network are restricted to use only a set of allowed domains (e.g., in some critical infrastructures), or if the resolution platform is restricted to using only domains that a nameserver is authoritative for (see discussion in Section III-A). To that end, we design indirect egress caches discovery and enumeration techniques, using timing side channels. The timing channel measures the difference in latency for cached vs. non-cached records.

In Sections IV-B1 and IV-B2 we design methodologies assuming a direct egress access and in Section IV-B3 we adapt the techniques for an indirect egress access.

1) Direct Ingress and Direct Egress Access: Open recursive resolvers provide direct access to triggering DNS requests at ingress resolvers. In particular, in order to discover and enumerate the caches used by a given ingress IP address of an open resolver on a resolution platform, our direct prober (set-up 2 in Figure 1) sends q queries to a resource record within our domain, such that all of the q queries are for *the same query name*. In the zone file of our domain `cache.example` we setup a corresponding DNS record, mapping the resource to an IP address, as follows: `name.cache.example IN A a.b.c.d`.

a) Caches Enumeration: On our nameserver we count the number of queries that arrive from an egress IP address of the target resolution platform for the name that our client requested. The number of queries $\omega < q$ arriving at our nameserver is the number of caches used by the resolution platform.

How many queries should be sent to an ingress IP address to guarantee that all the caches are probed? If the number of caches n is greater than q, we underestimate the caches. Alternately, using $q > n$ allows to cover all the caches. But, what should the value q be? In Section V-B we provide an analysis for the number of queries needed to probe all the caches behind an ingress IP address of a resolution platform. In Section V we describe the approach that we devised and used in our Internet evaluations.

b) IPs to Caches Mapping: In order to discover the mapping between $\{I_{IN}\}$ ingress IP addresses and clusters of caches we use the following approach: (1) we apply the caches enumeration technique (above) using any ingress IP address I^1_{IN} (out of a set of $\{I_{IN}\}$ ingress IP addresses), and plant a 'honey' record in all the caches mapped to that IP address. Then, for each ingress IP I^i_{IN} (for $1 < i < |\{I_{IN}\}|$) we send queries for the seeded 'honey' record. If queries are responded without accessing our server, we add I^i_{IN} to the same cluster of caches as I^1_{IN}. We perform this for every ingress IP until all are mapped.

The mapping from the set of caches to the egress IP addresses is straightforward: typically, different egress IP addresses participate in resolution chains. By repeating the experiment with a set of queries to an ingress IP address, and checking which egress IP addresses they arrive from at our nameservers, all the egress addresses can be covered. The analysis for number of queries is similar to ingress IP addresses and is given in Section V-B.

In contrast to open resolvers, when DNS resolution platforms are studied using email servers or web browsers, there is no direct access to the resolver, and all the queries are triggered by the (stub) DNS software. As we discussed above, local caches pose a challenge and impose two main limitations: same query will be responded from the local cache (without reaching the ingress resolver) and we cannot control the timing of the issued DNS requests; We next show two ways we devised to bypass the local caches.

2) Indirect Ingress and Direct Egress Access: In this section we show how to adapt our methodologies when an indirect access to ingress resolver is provided.

a) Bypassing Local Caches with CNAME Chain: We setup q DNS records in our `cache.example` zone mapping them to CNAME DNS record as follows:

```
x-1.cache.example   IN CNAME name.cache.example
x-2.cache.example   IN CNAME name.cache.example
...
x-q.cache.example   IN CNAME name.cache.example
name.cache.example  IN A     a.b.c.d
```

Then we trigger q DNS requests via email server or web browser, for names `x-1.cache.example`,..., `x-q.cache.example`. The local caches are not involved in the resolution process (specifically in resolving the CNAME redirection) and only receive the final answer.

b) Bypassing Local Caches with Names Hierarchy: This technique utilises the hierarchy that can be created with DNS names. We set up two zones as follows:

```
;zone fragment for sub.cache.example
$ORIGIN sub.cache.example.
x-1.sub.cache.example   IN A a.b.c.e
x-2.sub.cache.example   IN A a.b.c.e
...
x-q.sub.cache.example   IN A a.b.c.e
sub.cache.example       IN NS ns.sub.cache.example


;zone fragment for cache.example
$ORIGIN cache.example.
sub.cache.example       IN NS ns.sub.cache.example
ns.sub.cache.example       IN A a.b.c.d
```

Then we trigger q queries asking for A records of `x-1.sub.cache.example`,...,`x-q.sub.cache.example`. First time, the cache in the target resolution platform will ask for `cache.example`. The nameserver authoritative for `cache.example` will return a referral response for `sub.cache.example`, i.e., an NS and an A records for `ns.sub.cache.example`. When queried for an A record of `x-i.sub.cache.example` ($0 < i \leq q$), the nameserver will respond with an IP address `a.b.c.e`. During the subsequent queries, the cache will have stored the NS and A records for `sub.cache.example`, and should query it directly for the A records of `x-i.sub.cache.example`. The number of queries arriving at the nameserver of `cache.example` indicate the number of caches used by a given IP address at a measured resolution infrastructure.

3) Indirect Egress Access: When the CDE infrastructure is limited to using domains not under its control, the queries do not arrive at our nameservers and we cannot study the caches by monitoring the queries. To that end, we devise a timing side channel which allows counting the number of caches without observing the arriving queries. We force all the caches to store a honey record (in one of the domains that they can access) utilising sufficient redundancy to ensure that all caches are covered, e.g., issuing 100 queries to the resolution platform.

Assuming we have a direct ingress access, we measure the latency it takes the target resolution platform

to respond to queries for the honey record (that already exists in caches) vs records that are not in caches (e.g., a honey record with a random subdomain prepended to it). Specifically, our direct prober measures the latency of the responses that it receives.

When an indirect ingress access is provided, the study depends on locating domains with a structure similar to those described in Section IV-B2. Fortunately, such zone files are common. We perform a measurement evaluation for domains that use the names hierarchy (described in latter approach for caches bypassing) - popular Alexa domains that are under `com` TLD. For comparison between direct and indirect ingress access, we focus on the 1K domains where the nameserver IP addresses are hidden behind recursive resolution caches (see Section for details III-A), and use an open recursive resolution access to those domains as well as access via a web browser. We measure the latency it takes the resolution platform to respond to queries for the honey record when it is responded from the cache vs when the query is forwarded to a nameserver, and count the number of times the latency of the response that arrives at our prober corresponds to an uncached latency – this number corresponds to the amount of caches.

V. Measuring Resolution Platforms

In this section we apply the techniques developed in Section IV for study of DNS resolution platforms in the Internet. We first report on our results of enumeration of caches behind IP addresses and on correlation between IP addresses and caches. Then we provide an analysis on the bound of the number of packets required for our study of caches discovery and enumeration.

During our Internet measurements we incurred packet loss in some networks, which impacted the results. Highest packet loss was measured in Iran with 11%, China almost 4%; the rest networks exhibited around 1% packet loss which is considered typical. A lost packet during the tests affects the results of the test, and to cope with packet loss we use a statistical approach we dub *carpet bombing*, which is less sensitive to packet loss. The idea is intuitive we increase the number of probes we send to the target network, and instead of a single query we use K queries; such that the parameter K is a function of a packet loss in the measured network.

A. IP Addresses to Caches Mapping

Our measurements show that in very few cases the resolution platforms in the Internet use a single IP address, while typical platforms have multiple ingress or egress IPs. In Figure 3 we plot our measurements for number of egress IP addresses supported by resolution platforms of the three common networks' populations

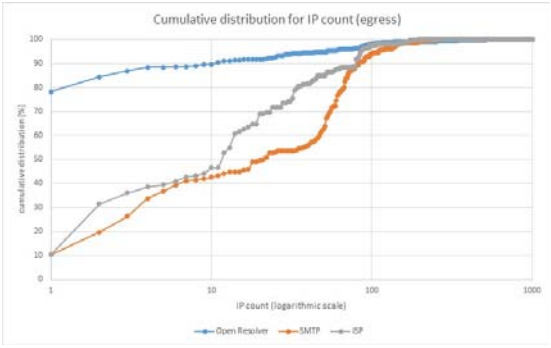

Fig. 3. The number of egress IP addresses supported by resolution platforms.

that we studied. In enterprises measured via email servers, 50% of the platforms use more than 20 IP addresses. In ISPs resolution platforms (measured via ad-network) 50% use more than 11 IP addresses. In networks operating open resolvers the situation is slightly different, 85% use 5 or less IP addresses.

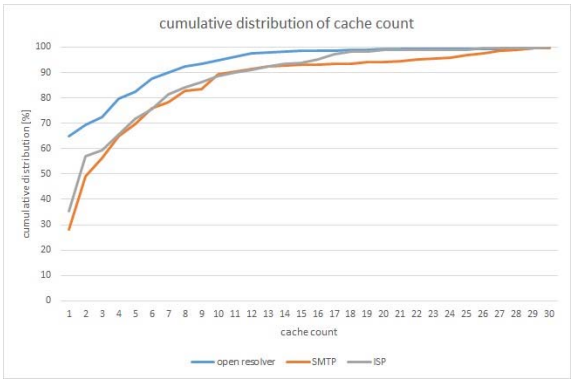

Fig. 4. The number of caches supported by resolution platforms.

Figure 4 shows our measurements of the cumulative number of caches in DNS resolution platforms. The networks running open resolvers use the least number of caches, 70% use 1-2 caches per IP address. About 60% of DNS platforms operated by ISPs use 1-3 caches, and 65% of networks measured via email servers use 1-4 caches per egress IP address.

The results in Figures 5, 7 and 8 provide the distribution of the number of ingress IP addresses vs. caches for networks operating open resolvers, enterprise networks and ISPs. The circles' sizes correspond to the number of measured networks that fall within that set, i.e., the larger the circle is the more networks call within that set. The center of the circle corresponds to the (x, y) coordinate on the graph. The majority of the networks with open resolvers have similar properties:

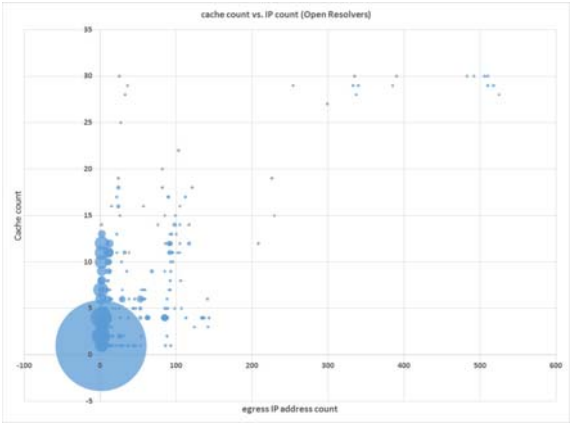

Fig. 5. IP addresses vs. caches count in DNS platforms with open resolvers.

use 1 ingress IP address and 1 cache – this corresponds to the largest circle in Figure 5. Smaller circles on y axis show that many other networks have less than 10 IP addresses. On the other hand, very few networks also use more than 500 IP addresses, with more than 30 caches (top right circles in Figure 5).

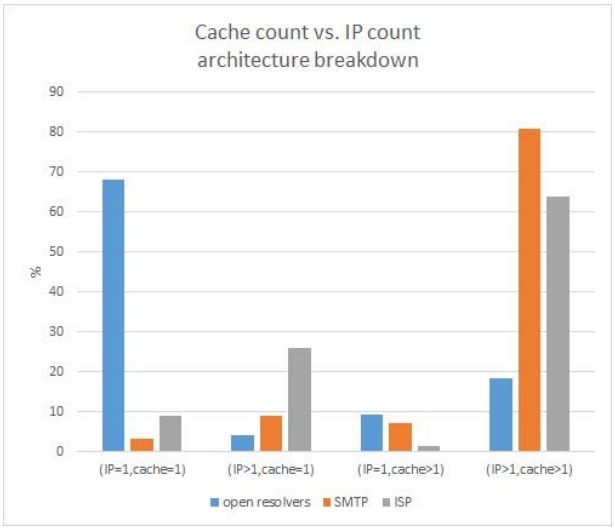

Fig. 6. IP addresses vs. caches count across three networks populations: open resolvers, enterprises and ISPs.

In contrast, the results for enterprise networks and networks of ISPs are more scattered, with a more even distribution and significantly less IP addresses. ISP networks appear to use least caches and have the smallest number of IP addresses, Figure 8.

We illustrate the percentages for different cache to IP ratio in Figure 6. Almost 70% of networks with open resolvers use DNS resolution platforms with one IP address and one cache. In contrast, less than 10% of ISP networks and less than 5% of enterprises use a single address and cache. The majority of ISPs and of enterprise networks use more than one address and more than one cache (almost 65% of IPSs and more than 80% of enterprises).

The difference in results between the networks with open resolvers and the enterprise and ISP networks corresponds to our data collection of networks of open resolvers. Specifically, these DNS resolution caches are configured to reduce traffic to the nameservers, to protect the nameservers against attacks and to lower the latency for clients' communication to the services. Since these caches are used only by clients accessing that specific domain, as well as by other services in those networks, the traffic volume is not high, hence a few or even a single cache suffices.

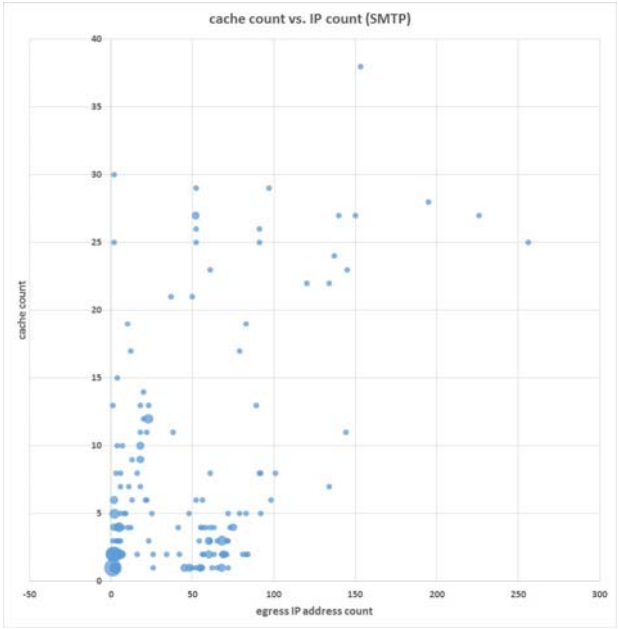

Fig. 7. IP addresses vs. caches count in SMTP population.

B. Analysis

In this section we provide bounds on the complexity of caches enumeration problem. The complexity depends on the cache selection algorithm, and on the traffic from other clients, arriving to the resolution platform.

Assuming a *round robin* cache selection and no traffic from other sources, then $q = n$ DNS requests would be needed to probe all the caches behind a given IP address in the resolution platform.

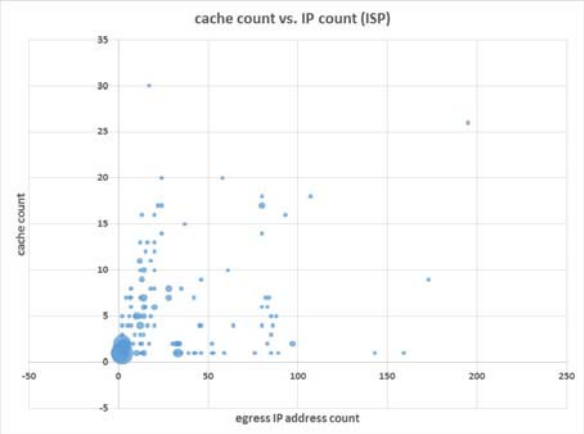

Fig. 8. IP addresses vs. caches count in ad-network population.

What is the expected number of DNS requests, needed to enumerate the caches assuming unpredictable cache selection strategy?

Caches enumeration is an extension of the combinatorial coupon collector problem [9], which one needs to collect all coupons to win in a contest. Specifically, given an urn of n different coupons from which coupons are being collected equally likely and with replacement, what is the probability that more than t sample trials are needed to collect all n coupons? In our setting the urns are caches, and the question is how many DNS queries does it take to make sure we cover all the caches?

Assume that each cache, out of n caches, is equally likely to be selected as a candidate for a given query. In each iteration exactly one cache is probed, and the experiment has to be repeated until each of n caches has been probed at least once. The cache selection is an independent random variable, and a cache i is selected with probability $p_i = \frac{1}{n}$. We then consider the following problem: *What is the expected number of queries q that needs to be issued in order to probe each of n caches?*

Theorem 5.1: Let X be the random number of queries that need to be issued in order to probe all n caches, such that $X = X_1 + ... + X_n$ and each X_i $(\forall 0 < i \leq n)$ denotes the number of queries required to probe cache i, after probing cache $i-1$. Then, the expected number of queries $q = E(X)$ to probe all n caches is:

$$E(X) = E(X_1) + ... + E(X_n) = \frac{1}{p_1} + ... + \frac{1}{p_n} = \frac{n}{n} + ... + \frac{n}{1}$$
$$= n \times \left(\frac{1}{1} + ... + \frac{1}{n}\right) = n \times \sum_{i=1}^{n} \frac{1}{i} = n \times H_n = n(\log n + \Theta(1))$$
$$= n \log n + O(n) = \Theta(n \log n)$$

Proof 5.2: The first cache is probed with first query, i.e., $X_1 = 1$. The probability to probe cache $i+1$, after probing i caches for $i \in \{1,...,n\}$, is $\frac{n-i}{n}$. Since the caches are selected with uniform probability X_i has a geometric distribution with parameter $\frac{n-i+1}{n}$. By linearity of expectations, we obtain that the expected number of queries to probe all the caches is $E(X) = E(X_1) + ... + E(X_n) = n \times \sum_{i=1}^{n} \frac{1}{i} = n \cdot H_n$, where H_n is the harmonic series, $H_n = \sum_{i=1}^{n} \frac{1}{i}$. For $n \to \infty$, the series converges to $H_n = \log n + \varepsilon + \frac{1}{2n} + O(\frac{1}{n^2})$. Hence $E(X) = n \log n + n\varepsilon + \frac{1}{2} + O(\frac{1}{n})$. We obtain $E(X) = n \log n + O(n) = \Theta(n \log n)$. ▩

In our Internet measurements, we perform caches enumeration in two phases: `init` and then `validate`, which we run N times in "parallel". Namely, during the initialisation phase we send N seeds: $s_1, s_2, ..., s_N$ (in parallel or in rapid succession). Then, we run the validate phase requesting for the records inserted during the `init` phase: $c_1, ..., c_N$ (in parallel or in rapid succession). During the validate phase we check for presence of our seeds in the caches, and count the instances of caches.

A prerequisite is that N (number of copies) is larger than n (number of caches). Specifically, the expected part of the n caches that is not covered in N attempts is roughly $exp(-\frac{N}{n})$, so only a small fraction of caches may be missed with $N = 2*n$. Consequently, with seeds $s_1, ..., s_N$ we statistically initiate all the caches and we count the caches $c_1, ..., c_N$ by checking for presence of seed records in them planted during the `init` phase. We expect success rate of $N \cdot (1 - exp(-\frac{N}{n}))^2$; as $\frac{N}{n}$ grows, this asymptotically reaches N.

VI. Related Work

Our work is related to the research on DNS platforms and its results and tools are especially important for security of DNS. We review related work and the relevant threats on caches and explain the relationship to our work.

Particularly relevant to our work are studies on the client side of the DNS infrastructure, including studies of the actors involved in resolution platforms and studying of DNS software fingerprinting. Prior work studied the impact of caching in the resolvers on DNS performance and on the latency perceived by the clients, e.g., [10], [11]. To optimise content distribution networks (CDNs) [12] ran a study of associating DNS resolvers with their clients, and also designed approaches to fingerprint the operating system or DNS software but did not evaluate them in the Internet-scale. This work was extended by [13], which fingerprinted a limited set of DNS software (Bind, Unbound and Windows) in the wild. Both works [12], [13] use patterns in DNS queries to fingerprint DNS software; example

patterns include the maximal queried length of `CNAME chains`, presence of requests for `AAAA` (following requests for `A` records). These techniques identify the software supported by the *egress IP addresses* of DNS resolvers, since they observe the query pattern which is independent of the caching behaviour.

The caches play a much more significant role in resolution chains than the ingress or egress resolvers. In particular, the ingress resolvers relay queries from clients to the caches, without applying caching logic on the received records. Ingress resolvers are also often configured to use upstream caches, such as Google Public DNS, in which cases the client will only see the forwarder whose sole functionality is to relay queries, while the complex caching logic is performed by the upstream cache. Furthermore, as we witnessed during our Internet evaluations, typically multiple IP addresses are involved in a resolution chain of a domain name, and hence different IP addresses perform the resolution. For instance, when resolving `www.foo.example`, the request to the parent domain `example` arrives from one egress IP address, and a subsequent request to `foo.example` arrives from a different egress IP address (more IP addresses are involved in longer resolution chains, e.g., such as `CNAME` chains). Consequently a DNS software running on an egress IP address is not representative of a DNS resolution platform and in particular, does not reflect the inner workings of the caching.

A study by [14] suggested to remove the DNS resolution platforms, and to leave the resolution to end hosts, arguing that the overhead on the existing end hosts would not be significant. [15] evaluated impact of domain name features on the effectiveness of caching. Recently, Schomp et al [16], measured the client side of the DNS infrastructure of *open recursive resolvers*, in order to identify all the actors in DNS resolution platforms. The goal of Schomp et al was to understand the actors involved in DNS resolution, however, their study did not cover the discovery and study of caches and the mappings between IP addresses and the caches hidden behind them. We extend the client-side DNS infrastructure studies with designing and evaluating methodologies for inferring the caches topologies and structures, and provide methodologies for inferring caches to IP addresses mapping, and for calculating the number of caches behind a given IP address of a DNS resolver.

A number of other studies were conducted on open resolvers, e.g., [17], [18], where the IPv4 address block is scanned for hosts responding to requests on port 53. However, recently it was shown by [19], [20] that most such open resolvers are either (misconfigured) home routers and mismanaged (security oblivious) networks or malicious networks operated by attackers (where the open DNS resolver is set up for malware communication to the command and control servers). In contrast to studies on open resolvers, our research is done on well managed networks, including enterprises, public DNS operators, Internet Service Providers (ISPs) and popular networks. We categorise the networks in our dataset and describe our data collect in Section III.

Security of DNS typically refers to correctness of DNS responses (against DNS cache poisoning attacks), and privacy of clients and systems (against censorship and monitoring).

There is a long history of DNS cache poisoning, [21], [22], [7]. DNS cache poisoning attacks are known to be practiced by governments, e.g., for censorship [23] or for surveillance [24], as well as by cyber criminals for credentials theft, malware distribution, and more. In the course of a DNS cache poisoning attack, the attacker provides spoofed records in DNS responses. These spoofed records are cached, and then returned to clients and systems, thereby redirecting them to incorrect hosts. The recent wave of cache poisoning vulnerabilities as well as the evidence for DNS injections in the Internet stimulated awareness within the operational and research communities, and a number of studies measuring DNS injections in the wild were conducted, [25], [17], [26]. In our recent work [8] we analysed vulnerabilities of caches to different records' injection methods. Another study [27] measured misconfigured domains with dangling records, and showed attacks exploiting them.

The results and conclusions derived from our study contribute to the design of unilateral, non-cryptographic defences against DNS cache poisoning. Although cryptographic defence with DNSSEC was standardised already two decades ago, [RFC4033-RFC4034], it is not widely deployed and even signed domains may often be vulnerable to attacks, [28]. Utilising multiple caches with unpredictable caches' selection strategy increases the difficulty for successful cache poisoning attacks. In particular, to be effective the cache poisoning attack has to be conducted against a specific cache. When multiple caches are used in a DNS resolution platform, for cache poisoning to succeed even a strong Man-in-the-Middle (MitM) attacker would need to generate large traffic volumes to be able to hit the same cache – which would lead to detection of the attack.

VII. Conclusions

DNS has evolved into a complex infrastructure with hosts receiving queries from the clients, caches that store the requested records and hosts communicating with the nameservers. We study the caches in popular

networks in the Internet, and developed tools for caches enumeration and mapping to the ingress and egress IP addresses.

Understanding and characterising the caches in networks in the Internet is primarily important for security, we provide few examples in our work. Multiple caches, along with unpredictable cache selection strategy, can significantly raise the bar for DNS cache poisoning.

Our study shows that the IP addresses provide one aspect of DNS resolution platforms, that is visible to the communicating parties, such as clients and nameservers. However, the IP addresses expose little information about the internal configurations in DNS resolution platforms. In particular, although in some cases we witnessed a one-to-one correspondence between IP addresses and caches, in most cases the IP addresses had little meaning. For instance, we often saw that multiple *different* egress IP addresses participated in a resolution of a given name, e.g., a `CNAME` chain often begins with one IP address, which is replaced by others in subsequent links in a `CNAME` chain.

VIII. Acknowledgements

The research reported in this paper has been supported in part by the German Federal Ministry of Education and Research (BMBF) and by the Hessian Ministry of Science and the Arts within CRISP (www.crisp-da.de/). This work has been co-funded by the DFG as part of project S3 within the CRC 1119 CROSSING. We are grateful to Microsoft Azure Research Award, which enabled us to host our infrastructure on Azure platform.

References

[1] A. Klein, "BIND 9 DNS cache poisoning," Trusteer, Ltd., 3 Hayetzira Street, Ramat Gan 52521, Israel, Report, 2007.

[2] A. Herzberg and H. Shulman, "Security of patched DNS," in *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*, 2012, pp. 271–288.

[3] H. Shulman and M. Waidner, "Towards security of internet naming infrastructure," in *European Symposium on Research in Computer Security*. Springer, 2015, pp. 3–22.

[4] ——, "Fragmentation Considered Leaking: Port Inference for DNS Poisoning," in *Applied Cryptography and Network Security (ACNS), Lausanne, Switzerland*. Springer, 2014.

[5] A. Herzberg and H. Shulman, "Vulnerable delegation of DNS resolution," in *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, 2013, pp. 219–236. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40203-6_13

[6] ——, "Socket Overloading for Fun and Cache Poisoning," in *ACM Annual Computer Security Applications Conference (ACM ACSAC), New Orleans, Louisiana, U.S.*, C. N. P. Jr., Ed., December 2013.

[7] ——, "Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org," in *IEEE CNS 2013. The Conference on Communications and Network Security, Washington, D.C., U.S.* IEEE, 2013.

[8] A. Klein, H. Shulman, and M. Waidner, "Internet-Wide Study of DNS Cache Injections," in *INFOCOM*, 2017.

[9] A. Boneh and M. Hofri, "The coupon-collector problem revisiteda survey of engineering problems and computational methods," *Stochastic Models*, vol. 13, no. 1, pp. 39–66, 1997.

[10] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "Dns performance and the effectiveness of caching," *Networking, IEEE/ACM Transactions on*, vol. 10, no. 5, pp. 589–603, 2002.

[11] D. Wessels, M. Fomenkov, N. Brownlee, and K. Claffy, "Measurements and laboratory simulations of the upper dns hierarchy," *Passive and Active Network Measurement*, pp. 147–157, 2004.

[12] C. A. Shue and A. J. Kalafut, "Resolvers revealed: Characterizing dns resolvers and their clients," *ACM Transactions on Internet Technology (TOIT)*, vol. 12, no. 4, p. 14, 2013.

[13] R. Chitpranee and K. Fukuda, "Towards passive dns software fingerprinting," in *Proceedings of the 9th Asian Internet Engineering Conference*. ACM, 2013, pp. 9–16.

[14] K. Schomp, M. Allman, and M. Rabinovich, "Dns resolvers considered harmful," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. ACM, 2014, p. 16.

[15] S. Hao and H. Wang, "Exploring domain name based features on the effectiveness of dns caching," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 1, pp. 36–42, 2017.

[16] K. Schomp, T. Callahan, M. Rabinovich, and M. Allman, "On measuring the client-side dns infrastructure," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 77–90.

[17] ——, "Assessing dns vulnerability to record injection," in *Passive and Active Measurement*. Springer, 2014, pp. 214–223.

[18] M. Kührer, T. Hupperich, J. Bushart, C. Rossow, and T. Holz, "Going wild: Large-scale classification of open dns resolvers," in *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*. ACM, 2015, pp. 355–368.

[19] D. Dagon, N. Provos, C. P. Lee, and W. Lee, "Corrupted dns resolution paths: The rise of a malicious resolution authority." in *NDSS*, 2008.

[20] J. Zhang, Z. Durumeric, M. Bailey, M. Liu, and M. Karir, "On the mismanagement and maliciousness of networks," in *to appear) Proceedings of the 21st Annual Network & Distributed System Security Symposium (NDSS14), San Diego, California, USA*, 2014.

[21] J. Stewart, "Dns cache poisoning–the next generation," 2003.

[22] D. Kaminsky, "It's the End of the Cache As We Know It," in *Black Hat conference*, August 2008, http://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Kaminsky/BlackHat-Japan-08-Kaminsky-DNS08-BlackOps.pdf.

[23] D. Anderson, "Splinternet behind the great firewall of china," *Queue*, vol. 10, no. 11, p. 40, 2012.

[24] M. Hu, "Taxonomy of the Snowden Disclosures," *Wash & Lee L. Rev.*, vol. 72, pp. 1679–1989, 2015.

[25] P. Levis, "The collateral damage of internet censorship by dns injection," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 3, 2012.

[26] M. Wander, C. Boelmann, L. Schwittmann, and T. Weis, "Measurement of globally visible dns injection," *Access, IEEE*, vol. 2, pp. 526–536, 2014.

[27] D. Liu, S. Hao, and H. Wang, "All your DNS records point to us: Understanding the security threats of dangling DNS records," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, 2016, pp. 1414–1425.

[28] H. Shulman and M. Waidner, "One Key to Sign Them All Considered Vulnerable: Evaluation of DNSSEC in Signed Domains," in *The 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX, 2017.