Towards Security of Internet Naming Infrastructure

Haya Shulman[1,2(✉)] and Michael Waidner[1,2]

[1] Fraunhofer Institute for Secure Information Technology (SIT), Technische Universität Darmstadt, Darmstadt, Germany
[2] Fachbereich Informatik, Technische Universität Darmstadt, Darmstadt, Germany
{haya.shulman,michael.waidner}@sit.fraunhofer.de

Abstract. We study the operational characteristics of the server-side of the Internet's naming infrastructure. Our findings discover common architectures whereby name servers are 'hidden' behind server-side caching DNS resolvers. We explore the extent and the scope of the name servers that use server-side caching resolvers, and find such configurations in at least 38 % of the domains in a forward DNS tree, and higher percents of the domains in a reverse DNS tree. We characterise the operators of the server-side caching resolvers and provide motivations, explaining their prevalence.

Our experimental evaluation indicates that the caching infrastructures are typically run by third parties, and that the services, provided by the third parties, often do not deploy best practices, resulting in misconfigurations, vulnerabilities and degraded performance of the DNS servers in popular domains.

1 Introduction

Domain Name System (DNS), [RFC1034, RFC1035], is the Internet's naming infrastructure; see background in Appendix, Sect. A. DNS plays a central role in the network operation, and its correctness and efficiency are critical to the stability and availability of the Internet. Initially designed to translate domain names to IP addresses, DNS infrastructure has evolved into a complex ecosystem and it is increasingly utilised to facilitate a wide range of applications. Due to the important function that DNS fulfills in the Internet, understanding and characterising it, is critical for security, efficiency and functionality of systems and networks. In this work we utilise Internet scale measurements to study the server-side of the DNS infrastructure. Within our study we find *common configurations* of DNS name servers, that utilise **server-side caching DNS resolvers to handle requests from the client-side resolvers**. In these configurations the DNS name servers are *hidden* behind recursive caching resolvers. In particular, the IP address of the server-side resolver is registered as the authoritative name server in the zone file of the target domain. As a result, client-side resolvers query that IP address (of the server-side resolver) and never communicate with the name

G. Pernul et al. (Eds.): ESORICS 2015, Part I, LNCS 9326, pp. 3–22, 2015.
DOI: 10.1007/978-3-319-24174-6_1

server directly (its IP address is not exposed to the client-side resolvers). In this work we identify and study the name servers supporting such configurations.

Caching constitutes an important building block in the design of scalable network architectures, and offers advantages such as an improved availability, security and reduced latency for responses to clients. Caching proxies are common on the client-side of the DNS infrastructure, where the DNS resolvers are connected to (often a chain of) caching forwarders. Such configurations are, however, much less known, and not studied, on the server-side. In our work, we find and study DNS configurations that use server-side caching resolvers, which handle the DNS requests, and relay them to the name servers. Our finding indicates that the server-side DNS architecture in the Internet is not limited to the traditional model, where the client-side resolution platform (i.e., a client-side resolver possibly connected via a chain of proxies) communicates with the name server directly (see Fig. 11 in Appendix, Sect. A). Our study also shows an increasing shift towards outsourcing DNS services' operation to expert third parties. Outsourcing services is an increasingly common practice in the Internet, [4]; it saves operational and management costs by using expertise and skilled personnel of the third party service provider, e.g., like the services provided by the cloud platforms. We characterise the operators of such caching server-side resolvers, and evaluate the security of domains using server-side caching resolvers. We discover severe vulnerabilities exposing to attacks both the misconfigured networks and other Internet victims. Our findings indicate that the hosts maintained by the third parties are often misconfigured recursive DNS resolvers, that do not support best practices and known security recommendations, e.g., [RFC5452, RFC4697]. Our study shows that many of the networks with the vulnerabilities and misconfigurations are *benign, popular* and often *security aware* – this is in contrast to anecdotal belief that misconfigurations are an artifact of security oblivious networks, [32]. We also explore the challenges that outsourcing DNS operations introduces for adoption of cryptographic defences for DNS.

Our study encompases domains in forward and reverse DNS trees. In a forward DNS tree we study 50K-top Alexa domains and Top-Level Domains (TLDs). In a reverse DNS we study the domains that correspond to IPv4 address blocks, i.e., all the network blocks in classes A, B and C – `x.in-addr.arpa.`, `y.x.in-addr.arpa.` and `z.y.x.in-addr.arpa.` respectively. Forward DNS typically hosts widely used services, such as web and email. Reverse DNS lookups are commonly utilised by security mechanisms, network operators and security researchers. For instance, domains in a reverse DNS tree are used to prevent spam and phishing attacks, to provide topological or geographic information of routers, or to prevent BGP prefix hijacking attacks.

We summarise our findings in Table 1. The first column (to the left), lists the DNS trees that we studied. The second column (to the left) contains the number of registered domains that we tested. This is mainly relevant in the reverse DNS tree, where a large fraction of the domains (that correspond to IPv4 address space) are not registered. The third column (to the left) contains the number of name servers in each domain space. The subsequent two columns report the name

Table 1. Summary of the results reported in this work.

DNS tree	Domains servers	Name	Fixed src port	Predictable src port	Failure w/DNSSEC	Server side cache	Open cache
Forward DNS Alexa	$50K$	32.5K	4 %	12.7 %	23 %	38 %	6 %
Forward DNS TLDs	568	3.2K	0.8 %	1.6 %	2 %	12 %	3.73 %
rDNS `x.in-addr.arpa.`	229	1.5K	7 %	14 %	2 %	14 %	8 %
rDNS `y.x.in-addr.arpa.`	$28K$	97K	10 %	19 %	32 %	41 %	19 %
rDNS `z.y.x.in-addr.arpa.`	$2,767K$	9,687K	14 %	19.5 %	34.5 %	38 %	21 %

servers with open server side resolvers that use fixed or predictable ports. Then we report on the number of servers that fail with DNSSEC enabled packets. The two rightmost columns contain the number of servers with server-side resolvers and with server-side resolvers supporting open recursive resolution respectively.

Organisation

This paper is structured as follows. We review related work and put our results in context in Sect. 2. In Sect. 3 we present a study of the server-side DNS infrastructure, and describe our methodology for detection of server-side DNS resolvers. In Sect. 4 we evaluate security of domains that use server-side resolvers, and conclude this work in Sect. 5. We provide an overview of DNS and DNSSEC in Appendix, Sect. A.

2 Related Work

In the following section we put our work in context with the related research. In particular, our work relates to prior studies of the: (1) DNS infrastructure, (2) misconfigured networks, and (3) DNS security.

2.1 Understanding the DNS Infrastructure

Studying the DNS infrastructure is important for design of Internet systems and future applications, and for construction and adoption of security mechanisms, including defences for DNS, e.g., against Denial of Service (DoS), [BCP38], or cache poisoning attacks, [RFC5452], and for defences that utilise DNS for authentication of services, such as IP prefixes authentication for routing security with `ROVER`, [6], or anti-spam mechanisms with `SPF`, [8].

A number of research works studied the client-side DNS infrastructure and vulnerabilities, e.g., [13,17,24,31]. On the server-side of the DNS infrastructure, [20,26] found multiple transitive trust dependencies within the DNS zones. Other work on the name server-side typically focuses on examining the DNS packets, exchanged between resolvers and name servers, such as for detection of malicious domains [1,5] or to detect incorrect uses of DNS, e.g., [3].

In this work we study the server-side *architecture* of the DNS infrastructure. In contrast to client-side resolvers, which the clients can identify by, e.g., inspecting the `hosts` file, the server-side resolvers are *transparent* to the clients, to the client-side resolvers and to network operators. This prevents the clients from being able to identify security vulnerabilities, or sources of failures, or tracking by third party DNS operators.

We also find that often, in contrast to best practices, [RFC5358, BCP140], the server-side third party resolvers support open recursive resolution, i.e., willing to lookup names in *any* domain and not only in the domain which they are 'authoritative' for.

We explore domains' configurations that use server-side resolvers, and illustrate the scope and the extent of this phenomenon in our work. We characterise the operators of the server-side DNS resolvers, and find that these are typically third party service providers.

2.2 Misconfigured Networks

Misconfigured networks pose a significant threat to the stability and availability of the Internet clients and services. Indeed, there is an established correlation between mismanagement and networks reponsible for malicious activities, [32]. Exploiting vulnerabilities in misconfigured networks is a stepping stone towards more sophisticated attacks and they are often abused by the attackers as *proxies* to attack victim networks and clients. Open recursive resolvers, willing to perform a recursive resolution for any Internet client, is a main source of the misconfigurations. Networks, operating open recursive resolvers, pose a particular threat, not only to the clients using their services, but also to the stability of the Internet, as they are frequently exploited in reflection amplification Denial of Service (DoS) attacks on victim networks and services, [22]. Unfortunately, despite the significant operational and research efforts to detect and characterise networks running open resolvers, [24,32], and to provide recommendations to mitigate the threat that they pose, the number of misconfigured networks is still overwhelming. One of the factors for this situation may be a common belief that since the misconfigured networks do not pose internal threat to their operators and clients, there is little incentive to fix the vulnerabilities, [32].

We identify one of the factors responsible for misconfigurations of networks: **outsourcing services to *security oblivious* third parties**. Although outsourcing network and services management to third parties can be effective and convenient, our results indicate that the security of these services should be carefully checked. We show experimentally that the vulnerable services are typically not hosted on the 'misconfigured' networks themselves, but on the networks which the third parties operate. Our findings also indicate that to optimise profit, third parties host multiple services of different customers *on the same hosts*. As a result, a vulnerability in one service, e.g., a web server, can be exploited to attack other services, e.g., DNS or email servers.

We hope that our work will raise awareness to the importance of validating security of the services provided by the third parties. Our message is that the

services provided by the third parties should not be blindly relied upon. Clients using third party services should validate the infrastructure of the third party service providers. We design tools that enable clients to infer information about the server side of the DNS infrastructure.

2.3 DNS Security

There is a long history of attacks against the DNS, most notably, DNS cache poisoning, [9,11,12,15,27,28]. In the course of a DNS cache poisoning attack, the attacker hijacks a victim domain by providing spoofed DNS records in DNS responses, thus redirecting clients to incorrect hosts, e.g., for credentials theft or malware distribution. As DNS plays an essential role in networks operation, cache poisoning can inflict economic losses and privacy damages and has a detrimental impact on the functionality and availability of the Internet clients and services. In particular, open recursive resolvers, that do not support source port randomisation and other recommendations, [RFC5452, RFC4697], are a lucrative target for cache poisoning attacks.

We find that often **the server-side resolvers do not support source port randomisation and use fixed or predictable ports for their requests to the name servers** (see columns 4 and 5 in Table 1). Resolvers with predictable ports are even more prevalent in domains in the reverse DNS tree, which is surprising since the reverse DNS is commonly utilised by the security mechanisms, hence it is expected to be better protected.

To mitigate the detrimental damages of cache poisoning attacks, IETF designed and standardised a cryptographic defence for DNS: DNSSEC [RFC4033-RFC4035]. A secure DNS would be resilient to cache poisoning attacks and would facilitate a wide range of applications and systems, such as secure routing (with `ROVER` [6]), secure email (with PGP keys distribution [29]).

Although proposed in 1997, DNSSEC is still not widely deployed; [18] found that less than 3 % of the DNS resolvers validate DNSSEC records. Important domains, such as the root and top-level domains (TLDs) are signed. However, the fraction of signed zones in lower domains, such as the second level domains (SLDs), is less than 1 %.

While there has been a considerable effort to identify the challenges to DNSSEC deployment, the focus was generally on the issues that large DNSSEC responses incur with the legacy firewalls and middleboxes on the resolver side, e.g., [14,30]. Zones signing was thought to be just a matter of motivation, and a folklore belief was that incentivised operators could sign their domains *'today'*.

Our study shows that the server-side resolvers impose obstacles on adoption of DNSSEC, and signing the zones would disrupt the DNS functionality to those domains. The reason is that the server-side resolvers often cannot operate with DNSSEC specific records or flags, [RFC4034-RFC4035], such as the `DO` bit in `EDNS` record, [RFC6891], and fail with an exception or a timeout; see the third column from the right, titled '`failure w/DNSSEC`', in Table 1.

In a recent work, [25], we studied security of encryption proposals for DNS, and showed that the intermediate proxies foil the security guarantees expected of the encryption schemes.

3 Studying DNS Name Servers

In this section we explore the architecture of the server-side DNS infrastructure. In particular, we answer the following questions: in Sect. 3.1 we consider the *what* - we identify common architectures of DNS name servers that use server-side caching DNS resolvers; in Sect. 3.2 we address the *why* - we study the advantages of such configurations; finally, in Sect. 3.3 we explore the *who* - we characterise the operators of server-side resolvers.

In Sect. 3.4, we introduce our methodology for detecting DNS name servers' architectures that use server-side DNS resolvers, and report on the extent and the scope of this phenomenon among popular domains in forward and in reverse DNS trees; our results and findings are summarised in Table 1.

3.1 Recursive Authoritative Name Servers

Our central finding is a common use of caches, that are configured to relay all the communication between the client-side resolvers and the name servers; see Fig. 1. In particular, we find that a large fraction of the domains in forward and reverse DNS trees are configured in the following way: an authoritative name server is *hidden* behind a server-side resolver. The IP address of the server-side resolver is reported as the name server in the zone file of the target domain (and in its parent). The server-side resolver receives DNS requests from the client-side resolvers and forwards them to the name server hosting the zone file for the target domain. Upon responses from the name server, the server-side resolver caches the DNS records and subsequently returns them to the requesting client-side resolver. Similarly to the standard DNS resolvers functionality, if the requested record is in the cache, the server-side resolver does not forward the query to the name server, but responds from the cache. The setting is illustrated in Fig. 1. We call the servers configured according to this setting the *recursive authoritative name server* (RANS). We report on the fraction of RANSes, among domains in forward and reverse DNS trees, in Table 1. The server-side resolver infrastructure may consist of a number of hosts, whereby a chain of resolver relay the request from one to another, until the request reaches the name server; see Fig. 2. We found that 42 % of the RANSes use more than one host in the server-side resolver infrastructure; we present the measurement methodology that we used in Sect. 3.4.

3.2 Why Use Server-Side Caches?

In this section we attempt to address the following question: **what are the reasons for such configurations?** We list the motivations for configuring caches 'before' the name servers, and compare to similar practices in other systems.

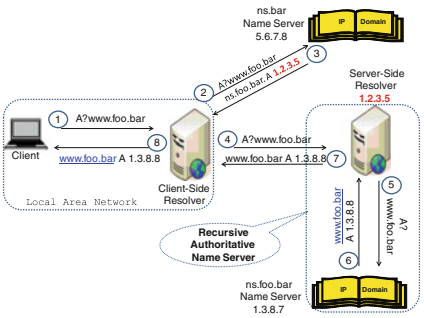

Fig. 1. Resolution for www.foo.bar hosted on a RANS.

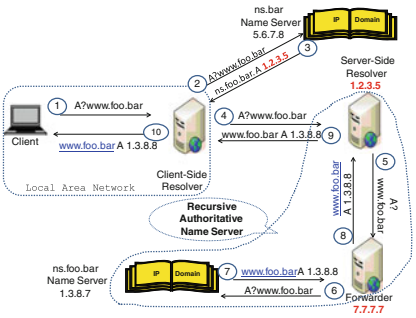

Fig. 2. Resolution to a RANS with a chain of server-side resolvers.

Improved Performance and Availability. To speed up access to their servers website operators utilise content delivery networks (CDNs), such as Akamai, these CDNs cache content in their global network and make it available to end users through geographically dispersed *edge servers* that are close to the users. Using CDNs for web content is a known practice. Our study shows that this practice is also common among the naming infrastructure. In particular, utilising distributed resolving hosts to retrieve, cache and supply DNS records to client can reduce latency for clients' requests – the IP address of the resolver can be ANYCAST-ed and the clients will perform lookups against the resolvers that are close to them.

Furthermore, if the name server is connected via a low bandwidth channel to the Internet or if its hardware is not suitable for handling multiple requests, a resolver with high bandwidth connectivity to the Internet or resolver's instance, distributed via ANYCAST, can solve this problem.

Enhanced Security. If the domain operator does not have the required expertise to enhance security of its name server, it can outsource this function to third party DNS operator. It is much more difficult to attack a hidden name server, since its IP address is not known. It is also much easier for the domain operator to prevent attacks, by configuring a firewall rule that allows requests to the name server only from one specific IP address – the resolver of the third party serving the (hidden) name server.

3.3 Who Operates and Uses RANS?

In this section we characterise operators of the server-side resolvers in RANSes, and the domains that use such configurations. To answer the former question we check the owner of the IP address of the server-side resolver, to answer the latter we check the owner of the domain of the name server.

Operators of RANSes. Running `whois` over the IP addresses of the server-side resolvers in RANSes, indicates that at least 57 % of the IP addresses belong to networks of commercial CDNs, [21], (such as Akamai, AT&T, NTT communication, LimeLight, Level 3 Verisign and Google). To identify CDNs, we used the

`traceroute` traces, and for each name server's IP address we checked whether `traceroute` from different locations yields different network adapters at the last host before the name server.

Typically, the caching service provided by the caching resolvers in a RANS configuration is purchased for a fee, however we also observed 'anecdotal' configurations, perhaps *free riders*, whereby domains' operators configure the open resolvers run by public open resolvers, e.g., such as Google Public DNS, as the IP addresses of the name servers of their domains.

Customers of RANSes. We use a domain query tool `DIG` to collect the set of `NS` records for RANS domains. Then, to characterise the domains with the RANS configuration, we perform the following tests over the `NS` names that we collected: (1) we check the age of each RANS domain, (2) we check the Alexa rank of the domains and (3) we check the type of domains.

These tests enable us to establish the 'reputation' of the networks which use the RANS configurations.

The age of the RANS domains can be used to establish domains' reputation. In particular, malware domains are typically characterised with a short lived duration (the attackers frequently change domain names to avoid detection) while a long lived domain can be used as an indicatation of legitimate operators. More than 50 % of the RANSes were registered on average a decade ago.

Many of the domains belong to financial institutions, university networks or Internet operators. These results show that many of the RANSes domains, are located in legitimate networks and run by legitimate (and not malicious) network operators.

Domain Age. Benign domains are usually characterised by a relatively long age. Domains used for malicious purposes instead are typically active only for short periods of time. The average age of benign domains is much higher than the average age of malicious domains. we have estimated that the average age of malicious hostnames is less than 5 weeks.

We utilise `whois` to find the average age of the RANS domains. The average age for domains in Alexa is 517 weeks, for a TLD the differences are very diverse: approximately 18 % are more than 20 years old, 11 % are between 3 and 5 years old, and 71 are less than 2 years old. In a reverse DNS tree, the average domain age is 972 weeks.

Alexa Rank. We examined the Alexa rank of domains running RANSes, and find that more than 60 % of the RANSes are ranked below 25,000 in Alexa; the results are plotted in Fig. 5.

Domain Type. We use a `whois` application over the RANS domains in forward and reverse DNS trees. We process the description field (*descr*) and organisation name (*org-name*) and check the values against a list of universities and network operators that we compiled. Those not appearing on the list, but with keywords such as *university, network, school*, were tested manually. We found that 38 % of the RANSes domains belong to educational sector, most notably universities, approximately 12 % belong to network operators, e.g., Sprint. Approximately

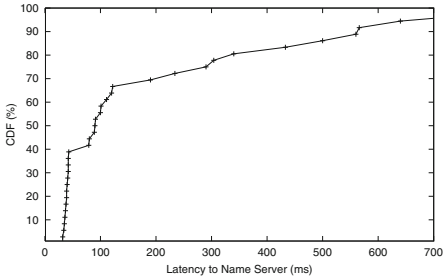

Fig. 3. Latency of the cached responses of the server-side resolver vs. responses from a (hidden) name server.

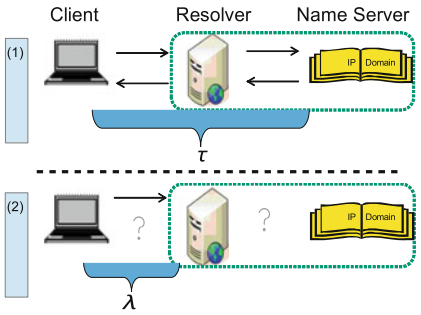

Fig. 4. Latency probing for the cached (λ seconds) vs. uncached record, i.e., including (τ seconds) latency to the name server.

18 % belong to domains in a financial sector. The rest could not be determined due to unknown or missing description or organisation name in information reported by the `whois` service.

3.4 Methodology for Detecting RANSes

We use the following techniques to identify domains with RANS architectures: (1) timing side channels, (2) Time-to-Live (TTL) and (3) open recursive resolution.

Timing Side-Channels. We design a timing side channel, which utilises the caching of the server-side resolvers, and allows to identify RANSes. The timing side channel is based on the difference in the latency of the responses relayed by the server-side resolvers to the (hidden) name servers, vs. the latency of the responses returned from the cache of the server-side resolvers. The timing channel is due to the caching of the resolvers. Our evaluation of a target domain `foo.bar` proceeds as follows: (1) we query a name server of `foo.bar` for a non-existing (random) subdomain `$str.foo.bar`, and measure the latency of the response; (2) we repeat the first step with the same query `$str.foo.bar` and measure the latency. If the name server uses a server-side resolver to handle the DNS queries for clients, then the latency of both responses will differ. In particular, since the record `$str.foo.bar` in the first query is not in the cache, it is relayed by the server-side resolver to the name server, a subsequent request for the same record is served from the cache. Hence the latency of the first request is higher than the latency of the second request – in this case we mark the name server infrastructure as a RANS. We illustrate the measurement technique in Fig. 4.

To account for a potential noise, such as occasional network load or load balancing mechanisms (where the query is sent to a different name server each time), we repeat the experiment for each name server 20 times, using a different random subdomain `$str` at each invocation.

The results of the RANSes measurements are plotted in Fig. 3. We set a threshold at 70 ms (and above), which is the typical delay in the Internet.

The values below 70 ms but above 30 ms, were marked as 'suspicious'. To confirm the 'suspicious' RANSes we used an additional TTL-based side channel (described below). As can be seen, the typical latency is above 100 ms. This significant difference in the latency is due to the fact that typically server-side resolvers and the (hidden) name servers are located in different Autonomous Systems (ASes).

Time-to-Live (TTL). Our measurement shows that server-side resolvers support a standard DNS caching mechanisms. Namely, upon receiving a response from the (hidden) name server, the records are sent to the requesting client-side resolver and are cached. The cache reduces the TTL value of the records, until it expires, and the records are evicted form the cache. We utilise the caching to identify server-side resolvers. The idea is to send a request for the same record twice (to the same name server) and check the TTL value of the record in both responses. If the TTL in the second response is lower than the TTL in the first response, then it is a RANS.

Our finding provides one possible explanation to the phenomenon of the inconsistency of TTL values in records within Alexa domains, as reported by [24]. In particular, [24] found that TTL of responses is distorted before reaching the requesting client, and that only 19 % of domains have consistent TTL values. We postulate that the bias could have resulted due to the requests to domains which use RANSes, see Fig. 1, and thus responses were served from the cache of the resolver.

The TTL side channel provides a more accurate metric, than the timing side channel, and enables to detect RANSes, where the latency between the server-side resolver and the (hidden) name server is not significant, e.g., below 30 ms. Although the TTL side channel is more reliable it nevertheless depends on the configuration of the proxy caching resolver. In particular, if the caching resolver does not accurately maintain the TTL of the cached records, the TTL channel cannot be relied upon. In contrast, the timing side channel, albeit less accurate, provides for a more reliable metric. For detection of server-side caching resolvers we recommend to utilise both side channels.

Open Recursion. We found that some of the RANSes are configured as *open resolvers*, i.e., the recursive resolution that they are willing to perform is not limited to a specific domain (for which they are registered as authoritative) but they will look up records in *any* domain.

We call these ORANSes. Open recursive resolution is known to expose to attacks and is considered a bad practice.

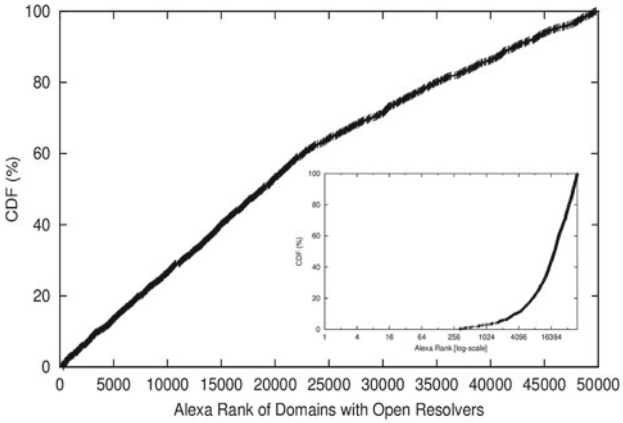

Fig. 5. Alexa rank of ORANSes; the inner graph presents results plotted in log-scale.

Blocking recursive resolution should also be enforced on the server side, and not only on the client side of the DNS infrastructure. On the server-side the goal is to restrict queries only for resources within the domain for which the target name server is authoritative, while serving queries from *any* clients. In contrast, on the client-side, the goal is to allow resolution for resources within *any* domain, but only for a limited set of clients, e.g., those that are located on the same network as the recursive resolver. To identify ORANSes, we set up a domain and sent requests for records within our domain, to the name servers, authoritative for the tested domains within forward and reverse DNS trees. Those servers that forwarded the requests to our name server were marked as ORANSes.

We identified 6 % of open RANSes in 50K-top Alexa domains, 3 % of open RANSes in TLDs, and we found a much larger fraction of the domains in a reverse DNS tree; the results are summarised in Table 1.

We discovered almost $3K$ ORANSes among top 50K Alexa domains and 21 ORANSes among TLDs. This translates to approximately 40 ORANSes in every 1K Alexa domains. We examined the Alexa rank of domains running ORANSes, and report our findings in Fig. 5.

To identify resolvers' chains, i.e., those that use forwarders, we concatenated the destination IP address (to which we sent the DNS request) as a subdomain of our domain; i.e., we sent DNS requests for `A` record of a resource in `dest-ip.our-domain.tld` domain, where `dest-ip` is the IP address of the server to which we sent the requests. This enabled us to associate the requests that we sent, with the requests that were subsequently received at our name server. Upon receipt of the request on our name server we validated if the request arrived at our name server from the same IP address, as the one to which it was sent. If not, we marked the ORANS as consisting of resolvers' chain. We found that 42 % of the RANSes are using a *chain* of recursive resolvers, consisting of at least two intermediate resolvers; see Fig. 2.

We found that 58 % of the ORANSes use the same source IP in the requests which they forward to the name servers, as the IP addresses on which they receive requests from the clients; namely, support the configuration illustrated in Fig. 1. In particular, 42 % of the DNS requests for `A` record of a resource in `our-domain.tld` domain, which we sent to the ORANSes, were received at our name server from different IP addresses.

We find that in 84 % of the requests, the IP address to which we sent the request, was located in a different AS than the IP address from which we received the request on our name server.

Misconfigurations. Almost 10 % of the RANSes in 25K-top Alexa domains return responses from a different IP address than the one to which the request was sent by the client-side resolver. In this case, the response (sent from an incorrect IP address of the name server) is ignored by the client side resolver, and after a timeout the query is resent to another name server.

4 Evaluating (in)Security of RANSes

In this section we evaluate the support of best practices and popular defences against cache poisoning by the server-side resolvers in the RANSes and ORANSes configurations.

4.1 Services Coresidence

We find that the server-side resolvers in RANSes frequently serve more than a single service. In particular, we tested for a 'coresidence' between web and DNS servers, in two phases: (1) we use `nmap` to check for open ports 80 or 443 on server-side resolvers in RANSes; then, in step (2) we use `telnet` to connect to the web server on port 80, and `s_client` of `openssl` to connect web servers that support communication over SSL/TLS, [RFC6101, RFC2246]. We find that more than 60 % of server-side resolvers host web and DNS services on the same machine. Hosting multiple services on the same machine is a known risky practice, in particular, a vulnerability in one service can enable attackers to take control over the host and subvert the security of the other services, e.g., vulnerability in PHP web servers enables attackers to obtain a shell on the victim host, [2].

4.2 Source Port Randomisation

Source port randomisation (SPR) is a main defence against DNS cache poisoning attacks. Resolvers supporting SPR send DNS requests from unpredictable (hopefully randomly selected) source ports. We tested support of SPR among server-side resolvers in ORANSes and RANSes. We first describe the sampling techniques that we used, and then analyse the values of the sampled ports.

Sampling SPR in ORANSes. We triggered, via each ORANS, four consecutive DNS requests to subdomains in our domain www.our-domain.tld, and capture the requests with a `tcpdump` on a name server hosting a zone file for our domain.

Sampling SPR in RANSes. The measurement of recursive resolvers in RANSes is tricky since we cannot trigger requests to a name server that we control – resolvers in RANSes are limited to resolving requests within a domain which they serve. The approach that we employ is based on a timing side channel due to packet loss inflicted by the attacker via socket overloading between hardware interrupts. The kernels in operating systems (OSes), e.g., Unix variants and Microsoft platforms, use hardware interrupts for event notification purposes in communication with input/output hardware components. Network interface cards (NICs) generate interrupts to notify the kernel of arrival of new packets. Hardware interrupts can impose significant CPU overhead. This is due to the fact that hardware interrupt is associated with context switching of saving and restoring processor state, see details in [16]. After the notification of a new packet arrival the kernel processes the packet, and then invokes TCP/IP protocol processing. However, *arrival of a new packet distrupts protocol processing* since hardware interrupts have higher priority over other tasks. Thus under a

high traffic load, the socket may fill up if the interrupt level is high, and when the socket queue is full the arriving packets will be dropped. Techniques to avoid socket overloading was studied in the scope of improving web servers efficiency, e.g., [19,23]. We use the socket overloading technique to elicit side channels for remote detection of port used by the resolver in a RANS. For our measurements, we employ 10 PlanetLab hosts to send the UDP packets' bursts. First we measure latency for requests to the resolver for records that are not in its cache (i.e., we concatenate a random subdomain to requests). Then, to sample if the resolver is using some port p, we send a burst of packets to a port p, causing socket overloading between kernel interrupts; this results in packets' loss. If the resolver used p to send its request to some name server, then the response from the name server to port p is also discarded; after a timeout (typically 1 second) the resolver retransmits the request. We use packet loss from the name server as an indication of hitting the correct port. Notice that a packet loss is caused only when the burst of packets is sent to the same port on which the resolver expects to receive a response. If a burst is sent to a different port, than the one from which the resolver sent its request, no loss will be incurred.

We find that packets of size 500 bytes provide for optimal packets' loss. This is probably due to the fact that they cause maximal number of hardware interrupts and filled the kernel buffers with bytes.

As a result, when the response from the sever arrived between the interrupts, it was discarded since the buffers were full. We first use the socket overloading technique above to sample for known fixed ports, and then for sequential ports. In case of sequential ports, we start with the highest port, probe each port 5 times, and then reduce the port by one, to next port; see Fig. 6.

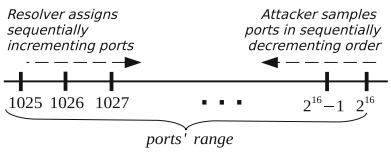

Fig. 6. Meet-in-the-middle port discovery procedure.

Analysing Port Selection. The results are plotted in Figs. 7, 8, and 9. As can be seen almost 10 % of the four consecutive requests contain a fixed port 53, and the similarity across the four requests is very high.

Among TLDs in forward DNS less than 1 % use a fixed port, and among 50K-top Alexa domains 4 % use a fixed port. In a reverse DNS domains that correspond to classes A, B and C – 7 %, 10 %, and 14 % respectively use fixed ports. The most popular (6.58 %) fixed port is 32,768, then follows a fixed port 53; see distribution of fixed ports in Fig. 7. Most of the requests that did not seem to use predictable ports, contained ports from small ports' ranges, which enables efficient exhaustive search of the ports pool. To calculate the ports' ranges, plotted in Fig. 8, we applied the following function over the ports in our four consecutive requests A, B, C and D: $(ABS(A-B)+ABS(C-D)+ABS(A-D))/3$ As can be seen, only less than 10 % of the requests have ports that differ by $17K$, while more than 90 % differ in less than 256 between them. In contrast, truly random ports would have an average difference of $2^{16}/2$, and would result in a normal distribution. Figure 9 plots the distribution of unpredictable ports in

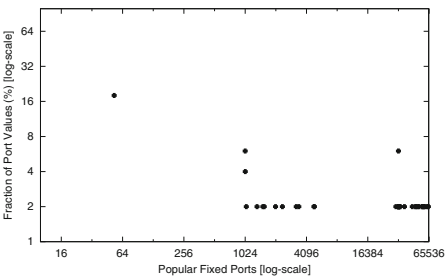

Fig. 7. Distribution of fixed ports in use by RANSes.

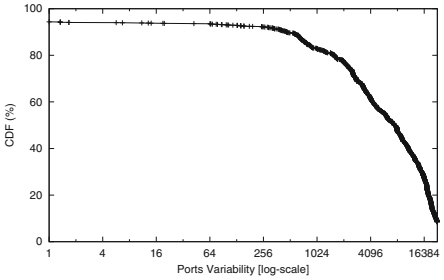

Fig. 8. Ports variability across different DNS requests.

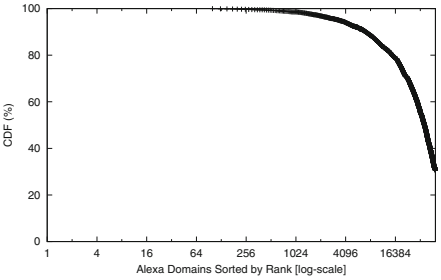

Fig. 9. Distribution of unpredictable ports within 25K-top Alexa domains.

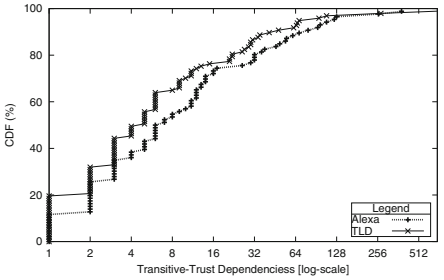

Fig. 10. Transitive-trust dependencies of domains in 50K-top Alexa and TLDs.

use by RANSes in 25K-top Alexa. The curve shows that port unpredictability is correlated with domain popularity – less popular domains use more predictable ports.

4.3 DNSSEC

The root zone and the top-level-domains (TLDs) are signed, but, as our measurements indicate, the adoption of DNSSEC within lower domains is extremely low; see Table 1. An interesting question is, how difficult is it to deploy DNSSEC, and whether there is software support for this. This question is particularly interesting in light of the coresidence of multiple domains, and in light of our discovery of recursive authoritative servers configurations. Our study provides a novel angle to the deployment of DNSSEC, allowing to look into the ability of name servers to support DNSSEC. In particular, previous work either measured the fraction of *signed* zones, [10,14,30], or the fraction of *validating resolvers*, [7,18], however, the ability and the readiness of the name servers to adopt DNSSEC or serve signed zones did not receive sufficient attention. We initiate this investigation in our work.

We sought to measure two quantities: (1) What fraction of open recursive authoritative name servers are 'DNSSEC compatible', i.e., can operate with

responses containing DNSSEC records. (2) What fraction of open recursive authoritative name servers perform strict validation of DNSSEC responses.

Evaluation Methodology. We set up three domains[1]: (1) without DNSSEC, (2) correctly signed with DNSSEC, (3) incorrectly signed. Domain (1) was not signed, and served plain DNS responses. Domains (2) and (3), were signed with 1024 bit keys RSA/SHA-1 (algorithm 5), [RFC3110,RFC4034]. Domain (3) served expired keys in `DNSKEY` records and invalid signatures in `RRSIG` record.

We found that 39.2 % of the open recursive authoritative name servers could not process signed DNSSEC responses, and returned `FMTERROR/SRVFAIL`, 29.8 % stripped DNSSEC records from responses and returned plain DNS responses (without signatures and keys). Together this resulted in a bit more than 69 % of servers that can not support DNSSEC. We found that only 30.9 % of the open recursive authoritative name servers return DNSSEC enabled responses, in return to requests for records in signed domains.

We further tested support of `EDNS0` among the open recursive authoritative name servers. Clearly, the 30.9 % of them that support DNSSEC also support `EDNS0`. What about the remaning 69 % that do not support DNSSEC. We found that 52 % of them support `EDNS0`. In total, we observed that 82 % of the open recursive name servers support `EDNS0` while 18 % do not.

To test whether open recursive authoritative name servers perform strict validation of DNSSEC responses, we ran test (2) on the 30.9 % of the open recursive authoritative name servers that could serve signed responses. None of the queries failed, namely, the open recursive authoritative name servers do not support strict DNSSEC validation. This result is consistent with measurements reported in [18], which showed that most recursive DNS resolvers, that support DNSSEC, do not perform strict DNSSEC validation.

4.4 Implications of Vulnerable RANSes

Vulnerabilities in RANSes can be exploited for large scale DNS cache poisoning attacks. In particular, in contrast to the tranditional cache poisoning, where only a single resolver, and the clients using it, fall victims – name servers using vulnerable third party caches expose *any* resolver querying that name server to attacks.

Such vulnerabilities in domains in a forward DNS tree expose to surveillance, distribution of malware, credentials theft, and more. Exploits in domains in a reverse DNS tree can be exploited to subvert security mechanisms, such as anti-spam defences, via `PTR` records, or routing security, such as `ROVER`. *Dependency on Vulnerable RANSes.* Best practices for ensuring availability and security of a domain in the DNS infrastructure recommend defining a number of name servers for each domain and configuring these name servers under at least two different parent domains. This redundancy provides for stability of the domain and prevents a single point of failure. In particular, if one of the parent domains is

[1] For compliance with anonymisation of the submission the domain names are removed.

not accessible, the domain will remain functional via the other parent domains. As a result of this practice, it is common in a DNS infrastructure for a domain name to depend on many other domains and resolving a single domain name often requires traversing multiple other domains. This phenomenon is called *transitive trust dependency*, introduced in [20]. On the flip side, while ensuring availability, this redundancy introduces a risk in case of dependencies on vulnerable domains. In Fig. 10 we plot the CDF of the number of transitive-trust dependencies in 50K-top Alexa and TLDs: 60 % of Alexa domains depend on 12 or more domains, and 60 % of TLDs depend on 6 or more domains. Subverting one of the domains in a transitive trust dependency chain would impact all the dependant domains.

5 Conclusions

In this work we performed a study of the server-side DNS infrastructure. Our results identify two common phenomena: (1) use of server-side caching DNS resolvers and (2) outsourcing security and availability of the domains to third party service providers. We study the implications of such architectures on the security, availability, and operational characteristics of popular domains in forward and reverse DNS trees, as well as on the design and adoption of security mechanisms for DNS. We show that often server-side resolvers do not support best practices and are vulnerable to cache poisoning attacks. In contrast to client side resolvers, where a vulnerability impacts only the clients on the network of the resolver, a vulnerability in a server-side resolver applies to *any* Internet client querying the vulnerable name server, and not limited to a specific network.

Since the server-side resolvers are transparent to the clients (client-side resolvers or network operators) querying them, the clients have no means to identify such architectures. Hence, the clients cannot validate whether they are vulnerable to cache poisoning attacks or identify failures, e.g., due to use of security mechanisms that the server-side resolvers cannot process, such as signaling of DNSSEC. We design a methodology and implement tools for detection of server-side resolvers and evaluation of their security. The results reported in this work, and the tools that we developed, are of interest and of importance for domain operators and clients, and enable automated evaluation of the security provided by third parties.

Acknowledgements. This research was supported by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE, by the Hessian LOEWE excellence initiative within CASED, and co-funded by the DFG as part of the CRC 1119 CROSSING.

A Overview: DNS and DNSSEC

Domain Name System (DNS) is composes of a client-server protocol, used by the resolvers to retrieve domain records in zone files maintained by the name servers.

The resolvers communicate to the name servers using a simple request-response protocol (typically over UDP); for instance, (abstracting out details) to translate www.foo.bar resolvers locate the name server `ns.foo.bar`, authoritative for `foo.bar`, and obtain the IP address of the machine hosting the web server of the website www.foo.bar, see Fig. 11. Resolvers store the DNS records, returned in responses, in their caches for the duration indicated in the Time To Live (TTL) field of each record set.

The resource records in DNS correspond to the different services run by the organisations and networks, e.g., hosts, servers, network blocks.

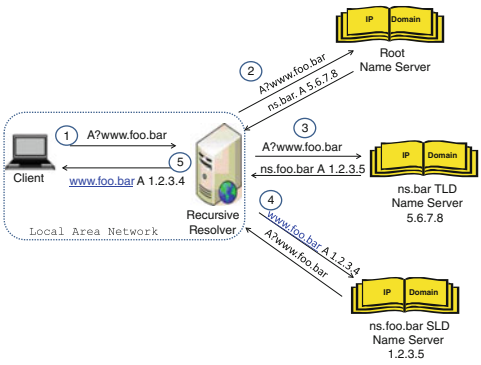

Fig. 11. DNS resolution process for www.foo.bar and the involved DNS servers.

The zones are structured hierarchically, with the root zone at the first level, Top Level Domains (TLDs) at the second level, and millions of Second Level Domains (SLDs) at the third level. The IP addresses of the 13 root servers are provided via the *hints* file, or compiled into DNS resolvers software and when a resolver's cache is empty, every resolution process starts at the root. According to the query in the DNS request, the root name server redirects the resolver, via a `referral` response type, to a corresponding TLD, under which the requested resource is located. There are a number of TLDs types, most notably: *country code TLD* (ccTLD), which domains are (typically) assigned to countries, e.g., `us, il, de`, and *generic TLD* (gTLD), whose domains are used by organisations, e.g., `com, org`, and also US government and military, e.g., `gov, mil`. Domains in SLDs can also be used to further delegate subdomains to other entities, or can be directly managed by the organisations, e.g., as in the case of `ibm.com, google.com`.

A DNS *domain* is divided into zones, and includes all the nodes of the subtree rooted at the zone. A DNS *zone* constitutes a portion of a domain name space. A zone contains only the nodes that are managed by the name server at the named node. A zone can be divided into subdomains, with its own DNS name servers. At the lowest level of the DNS tree, in the leaves of the tree, the terms 'DNS zone' and a 'DNS domain' become equivalent. For instance, when querying the

root zone for `foo.bar.`, the resolver will be redirected to `bar.` domain, via a `referral` to the authoritative servers for `bar.` zone. When querying the name servers of `bar.`, the resolver is issued another `referral` for `foo.bar.` zone. Notice that `bar.` zone does not include subdomains, e.g., like `foo.bar.`, but those are delegated from `bar` to their name servers.

When no protection is employed, DNS requests and responses can be inspected and altered by a MitM attacker. For example, a malicious wireless client can tap the communication of other clients and can respond to their DNS requests with maliciously crafted DNS responses, containing a spoofed IP address, e.g., redirecting the clients to a phishing site. Domain Name System Security Extensions (DNSSEC) standard [RFC4033, RFC4034, RFC4035] was designed to address the cache poisoning vulnerability in DNS, by providing *data integrity* and *origin authenticity* via cryptographic digital signatures over DNS resource records. The digital signatures enable the recipient, e.g., resolver, that supports DNSSEC validation, to check that the data in a DNS response is the same as the data published within the target zone.

DNSSEC defines new resource records (RRs) to store signatures and keys used to authenticate the DNS responses. For example, a type RRSIG record contains a signature authenticating an RR-set, i.e., all mappings of a specific type for a certain domain name. By signing only RR-sets, and not specific responses, DNSSEC allows signatures to be computed *off-line*, and not upon request; this is important, both for performance (since signing is computationally intensive) and security (since the signing key can be stored in a more secure location than the name server).

To allow clients to authenticate DNS data, each zone generates a signing and verification key pair, (sk, vk). The signing key sk is used to sign the zone data, and should be secret and kept offline. Upon queries for records in a domain, the name server returns the requested RRs, along with the corresponding signatures (in a RRSIG RRs). To prevent replay attacks, each signature has a fixed expiration date. The clients, i.e., resolvers, should also obtain the zone's public verification key vk, stored in a DNSKEY RR, which is then used by the clients to authenticate the origin and integrity of the DNS data.

Resolvers are configured with a set of verification keys for specific zones, called *trust anchors*; in particular, all resolvers have the verification key (trust anchor) for the root zone. The resolver obtains other verification keys, which are not trust anchors, by requesting a DNSKEY resource record from the domain. To validate these verification keys obtained from DNSKEY, the resolver obtains a corresponding a DS RR from the parent zone, which contains a hash of the public key of the child; the resolver accepts the DNSKEY of the child as authentic if the hashed value in DNSKEY is the same as the value in the DS record at the parent, and that DS record is properly signed (in a corresponding RRSIG record). Since the DS record at the parent is signed with the DNSKEY of the parent, authenticity is guaranteed.

This process constructs a *chain of trust* which allows the resolver to authenticate the public verification key of the target zone. Specifically, the clients

authenticate the public verification key of the zone by constructing a chain of trust starting at the root zone, or another trust anchor, and terminating at the target zone.

References

1. Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou II, N., Dagon, D.: Detecting malware domains at the upper dns hierarchy. In: USENIX Security Symposium, p. 16 (2011)
2. Canali, D., Balzarotti, D., et al.: Behind the scenes of online attacks: an analysis of exploitation behaviors on the web. In: Proceedings of the 20th Annual Network & Distributed System Security Symposium (2013)
3. Chen, Y., Antonakakis, M., Perdisci, R., Nadji, Y., Dagon, D., Lee, W.: DNS noise: measuring the pervasiveness of disposable domains in modern DNS traffic (2014)
4. Feamster, N.: Outsourcing home network security. In: Proceedings of the 2010 ACM SIGCOMM Workshop on Home Networks, pp. 37–42. ACM (2010)
5. Gao, H., Yegneswaran, V., Chen, Y., Porras, P., Ghosh, S., Jiang, J., Duan, H.: An empirical reexamination of global dns behavior. In: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, pp. 267–278. ACM (2013)
6. Gersch, J., Massey, D.: Rover: Route origin verification using DNS. In: 2013 22nd International Conference on Computer Communications and Networks (ICCCN), pp. 1–9. IEEE (2013)
7. Gudmundsson, O., Crocker, S.D.: Observing DNSSEC Validation in the Wild. In: SATIN, March 2011
8. Herzberg, A.: DNS-based email sender authentication mechanisms: a critical review. Comput. Secur. **28**(8), 731–742 (2009)
9. Herzberg, A., Shulman, H.: Security of patched DNS. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 271–288. Springer, Heidelberg (2012). http://dx.doi.org/10.1007/978-3-642-33167-1_16
10. Herzberg, A., Shulman, H.: DNSSEC: interoperability challenges and transition mechanisms. In: Eighth International Conference on Availability, Reliability and Security (ARES), 2013, Regensburg, Germany, pp. 398–405. IEEE (2013)
11. Herzberg, A., Shulman, H.: Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org. In: IEEE CNS 2013. The Conference on Communications and Network Security. Washington, IEEE (2013)
12. Herzberg, A., Shulman, H.: Socket overloading for fun and cache poisoning. In: Payne Jr., C.N. (ed.) ACM Annual Computer Security Applications Conference (ACM ACSAC), New Orleans, Louisiana, U.S., December 2013
13. Herzberg, A., Shulman, H.: Vulnerable delegation of DNS resolution. In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 219–236. Springer, Heidelberg (2013). http://dx.doi.org/10.1007/978-3-642-40203-6_13
14. Herzberg, A., Shulman, H.: Retrofitting security into network protocols: the case of DNSSEC. IEEE Internet Compu. **18**(1), 66–71 (2014)
15. Kaminsky, D.: 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
16. Kleiman, S.R.: Apparatus and method for interrupt handling in a multi-threaded operating system kernel, US Patent 5,515,538, 7 May 1996

17. Kührer, M., Hupperich, T., Rossow, C., Holz, T.: Exit from hell? reducing the impact of amplification DDoS attacks. In: Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, 20–22 August 2014, pp. 111–125 (2014)
18. Lian, W., Rescorla, E., Shacham, H., Savage, S.: Measuring the practical impact of DNSSEC Deployment. In: Proceedings of USENIX Security (2013)
19. Ramakrishnan, K.: Performance considerations in designing network interfaces. IEEE J. Sel. Areas Commun. **11**(2), 203–219 (1993)
20. Ramasubramanian, V., Sirer, E.: Perils of transitive trust in the domain name system. In: Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement, pp. 35–35. USENIX Association (2005)
21. Rayburn, D.: CDN market getting crowded: Now tracking 28 providers in the industry. Bus. Online Video Blog (2007)
22. Rossow, C.: Amplification hell: Revisiting network protocols for ddos abuse (2014)
23. Salah, K., El-Badawi, K., Haidari, F.: Performance analysis and comparison of interrupt-handling schemes in gigabit networks. Comput. Commun. **30**(17), 3425–3441 (2007)
24. Schomp, K., Callahan, T., Rabinovich, M., Allman, M.: On measuring the client-side DNS infrastructure. In: Proceedings of the 2013 Conference on Internet Measurement Conference, pp. 77–90. ACM (2013)
25. Shulman, H.: Pretty bad privacy: pitfalls of DNS encryption. In: Proceedings of the 13th Annual ACM Workshop on Privacy in the Electronic Society, WPES 2014, pp. 191–200 (2014). IETF/IRTF Applied Networking Research Award
26. Shulman, H., Ezra, S.: Poster: On the resilience of DNS infrastructure. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 1499–1501. ACM (2014)
27. Shulman, Haya, Waidner, Michael: Fragmentation considered leaking: port inference for DNS poisoning. In: Boureanu, Ioana, Owesarski, Philippe, Vaudenay, Serge (eds.) ACNS 2014. LNCS, vol. 8479, pp. 531–548. Springer, Heidelberg (2014)
28. Stewart, J.: DNS cache poisoning-the next generation (2003)
29. Wouters, P.: Using DANE to Associate OpenPGP public keys with email addresses (2014). http://tools.ietf.org/html/draft-wouters-dane-openpgp-02
30. Yang, H., Osterweil, E., Massey, D., Lu, S., Zhang, L.: Deploying cryptography in internet-scale systems: A case study on dnssec. IEEE Trans. Dependable Secure Comput. **8**(5), 656–669 (2011)
31. Yu, Y., Wessels, D., Larson, M., Zhang, L.: Authority server selection of DNS caching resolvers. ACM SIGCOMM Comput. Commun. Rev. **42**, 80–86 (2012)
32. Zhang, J., Durumeric, Z., Bailey, M., Liu, M., Karir, M.: On the mismanagement and maliciousness of networks. In: Proceedings of the 21st Annual Network & Distributed System Security Symposium (NDSS 2014), San Diego, California, USA (2014, to appear)