EURASIP Journal on Information Security
a SpringerOpen Journal

RESEARCH Open Access

DNSSEC for cyber forensics

Haya Shulman[1*] and Michael Waidner[1,2]

Abstract

Domain Name System (DNS) cache poisoning is a stepping stone towards advanced (cyber) attacks. DNS cache poisoning can be used to monitor users' activities for censorship, to distribute malware and spam and to subvert correctness and availability of Internet clients and services. Currently, the DNS infrastructure relies on challenge-response defences against attacks by (the common) off-path adversaries. Such defences do not suffice against stronger, man-in-the-middle (MitM), adversaries. However, MitM is not believed to be common; hence, there seems to be little motivation to adopt systematic, cryptographic mechanisms. We show that challenge-response do not protect against cache poisoning. In particular, we review common situations where (1) attackers can frequently obtain MitM capabilities and (2) even weaker attackers can subvert DNS security. We also experimentally study dependencies in the DNS infrastructure, in particular, dependencies within domain registrars and within domains, and show that multiple dependencies result in more vulnerable DNS. We review domain name system security extensions (DNSSEC), the defence against DNS cache poisoning, and argue that not only it is the most suitable mechanism for preventing cache poisoning but it is also the only proposed defence that enables *a posteriori* forensic analysis of attacks.

Keywords: DNS cache poisoning; Domain hijacking; Cyber attacks; Cyber security

1 Introduction

During the recent decade, the Internet has experienced an increase in sophisticated attacks, subverting stability and correctness of many networks and services. The attacks target individuals as well as enterprises and exploit vulnerabilities in systems and services in the Internet, e.g., WEB and cloud, as well as in basic building blocks of the Internet, such as Domain Name System (DNS), [RFC882,1034], and routing. The attacks inflict economical losses to businesses and have a devastating impact on e-commerce, security and critical infrastructure.

In this work, we focus on DNS, whose correctness and availability are critical to the functionality of the Internet. We investigate one of the most significant threats to DNS infrastructure: *cache poisoning*. In a cache poisoning attack, the adversary causes recursive DNS resolvers to accept and cache a spoofed DNS response which contains malicious records. These records redirect the victim clients to incorrect (possibly malicious) hosts. DNS cache poisoning is detrimental to correct functionality of Internet services and can be used to distribute malware and spam and can be applied for phishing attacks, credentials theft, eavesdropping and more.

To prevent DNS cache poisoning attacks, most systems adopt challenge-response mechanisms, [RFC6056, RFC5452], whereby a random challenge is sent within the request and a corresponding value is verified to have been echoed in responses. However, challenge-response authentication is not effective against *man-in-the-middle* (MitM) adversaries (Figure 1), which can inspect the challenges sent within the requests, and craft spoofed responses with valid challenge values. However, it is typically assumed that the common adversary in the Internet is *off-path*, which unlike a MitM, cannot observe nor modify legitimate packets exchanged between other parties.

Recently, a number of vulnerabilities were shown, allowing off-path attackers to predict values of challenge-response defences, exposing the recursive DNS resolvers to off-path cache poisoning attacks [1-6]. Some of these vulnerabilities were already patched, e.g., [7,8]. These attacks in tandem with the recent revelations on the surveillance programs, such as the ones carried out by the National Security Agency (NSA) [9,10], raise the question whether the widely deployed challenge-response defences

*Correspondence: haya.shulman@cased.de
[1]Fachbereich Informatik Technische Universität Darmstadt Mornewegstraße 32, Darmstadt DE, Germany
Full list of author information is available at the end of the article

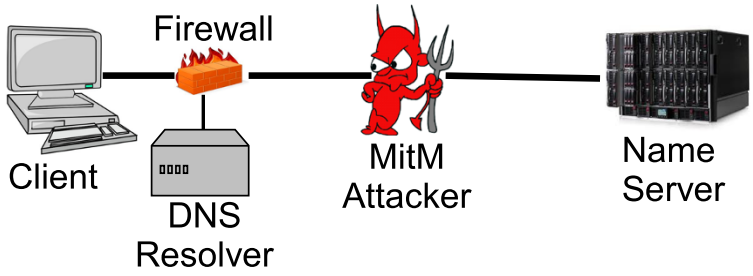

Figure 1 Man-in-the-middle attacker model.

suffice to ensure security of services and networks that rely on the correctness and availability of DNS.

We explore DNS security and review the threats that stem from the ubiquitious Internet connectivity, from the cyber arms race, advances in adversarial capabilities and from vulnerable systems. Our review of the recently published vulnerabilities, shows that systems, services and clients are vulnerable and may be frequently attacked. Widely deployed defences against off-path adversaries do not provide the adequate level of security required to thwart attacks by the modern adversaries. Dependencies within the DNS infrastructure, [11,12], further exacerbate the risk of attacks.

We believe that the main factor to this unfortunate situation is that the attacks that do not break connectivity go undetected - which is indeed the case with advanced, cyber and Advanced Persistent Threat (APT) attacks. For instance, recent hijacking of `google.rw` by the Syrian hackers [13] would go undetected had it not broken access to the target domains, and the surveillance by the US agencies would not be unveiled had it not been exposed by a whistleblower.

Our main message in this work is that domain name system security extensions (DNSSEC) is the most suitable defence for DNS against cache poisoning attacks. As we show, the significance of DNSSEC is not only in preventing cache poisoning (and thus other advanced) attacks but also in its ability to enable *detection* of attacks *a posteriori*. In fact, DNSSEC is the only mechanism that facilitates forensic analysis of attacks and provides evidences, which can be presented to third parties and which allow detection of attacks even by very strong adveraries, such as goverment agencies.

As we show in our study, DNSSEC is essential and critical for detection and prevention of attacks and protection of systems in light of the prevalence of sophisticated attacks by modern adversaries.

We next summarise the topics presented in this work.

1.1 MitM is common

Contrary to folklore belief, MitM adversaries are common. We review the cache poisoning attacks in the common settings (below) and discuss the adversarial capabilities required to launch them.

(1) *MitM on open access networks.* In the early days of the Internet, typical access of clients to Internet services was via their (trusted) home Internet service providers (ISPs), which, among others, provided recursive DNS resolution services. However, during the recent decade, we witness a growing adoption of IEEE 802.11 wireless networks [RFC5416] and an increasing number of clients accessing the Internet from public, often untrusted networks. This introduces new threats, which stem both from malicious network operators as well as from malicious insiders (other clients).

(2) *MitM on backbone links.* Powerful adversaries, controlling routers located on backbone links have access to all traffic, and recent revelations [10] show that there is a practice of injecting poisoned DNS responses into legitimate flows to redirect clients to incorrect servers, e.g., for censorship.

(3) *MitM via routing hijacking.* Inter-domain routing protocol, Border Gateway Protocol (BGP) [RFC4271], has a long history of route hijacking attacks [14], but it still does not employ cryptographic defences to guarantee routing correctness. Trivially, redirecting traffic via a different route or network can provide the attacker with MitM capabilities. Recently, such attacks were shown to be practical [15], and such hijacks may be frequently occurring.

1.2 Vulnerable name servers and registrars

Many of the DNS cache poisoing attacks occur by subverting a registrar or a name server. In contrast to the local nature of DNS cache poisoning attacks, which target a specific resolver, the impact of such attacks is global, i.e., any resolver receiving a response from the zone file hosted on a compromised server is a potential victim. We review recent attacks, along with the vulnerabilities that allowed them. We perform an evaluation of the vulnerabilities in domain registration interfaces (which registrars provide to customers).

1.3 Dependencies in Domain Name System

We review the operational characteristics of the DNS infrastructure: *transitive trust*, *coresidence* and *servers placement*. We argue that these factors impact resilience, stability and security of the DNS services. The high coresidence rate can disrupt services to multiple domains during benign failures or attacks on a single name server and high concentration of name servers in certain geographical locations can facilitate censorship of (and attacks on) a high volume of DNS requests.

1.4 Domain Name System security extensions

We believe that DNSSEC is the most suitable defence not only for thwarting cache poisoning attacks but also for detecting them. DNSSEC [RFC4033-4035] is a standard cryptographic protection for DNS that authenticates records via digital signatures. Although proposed and standardised in 1997, DNSSEC is still not widely deployed: most zones are not signed and most resolvers do not validate DNSSEC-signed responses.

Furthermore, early adopters experience failures and deployment problems. We review some notable failures and recommend automation of deployment as a mitigation. Our goal is to encourage deployment of DNSSEC, and we hope that our work will foster research efforts on the specific aspects which we identified as deterrents towards (correct) DNSSEC deployment.

We show that DNSSEC provides cryptographic evidences that can be used in forensic analysis and detection of attacks long after they occured, in particular even attacks launched by state entities, domain operators or MitM adversaries. This is in contrast to all other proposed defences for DNS, e.g., Eastlake cookies [16] or DNSCurve [17].

1.5 Contributions

In this work, we show that a critical system of the Internet, DNS, is vulnerable to attacks and that in contrast to folklore belief, strong attackers, such as MitM, are common. Attacks on DNS are detrimental for Internet clients and services.

We show that DNSSEC is the only standardised mechanism which can provide evidences for forensic analysis of attacks launched by the strong and sophisticated adversaries and can facilitate detection of attacks which would otherwise remain unnoticed.

Unfortunately, as our study indicates, the adoption of DNSSEC among zones in forward and reverse DNS trees is extremely low. Hence, further research is required to identify obstacles to DNSSEC deployment. We hope that our work will motivate such an investigation.

1.6 Organisation

We review DNS and DNS cache poisoning in Section 2. We discuss threats from (1) MitM adversaries and how attackers can obtain MitM capabilities (Section 3) and (2) vulnerabilities in name servers and zones hosting infrastructure (Section 4). We then review the dependencies within the DNS infrastructure in Section 5. We discuss DNSSEC and report on our measurements of DNSSEC adoption rate and deployment challenges. We then review application of DNSSEC for forensic analysis (Section 6). We conclude this work in Section 7.

2 Domain Name System and cache poisoning

In this section, we provide a background on DNS and provide a describe DNS cache poisoning. We discuss the phases of cache poisoning in detail.

2.1 Domain Name System

The Domain Name System (DNS), [RFC1024, RFC1025], is a distributed data base of Internet mappings (also called *resource records (RRs)*), from *domain names* to different values. For example, A type RRs map a domain name to its IPv4 address.

Domains are organised hierarchically; for every domain name α and each label or domain name x, the domain name $x.\alpha$ is considered a *subdomain* of α, i.e., part of the α domain name space. Namely, the rightmost label conveys the top-level domain.

Domains and their mappings are also administered hierarchically; the mappings of each domain *foo.bar* are provided by a *name server*, managed by the owner of the domain. The name server of a domain *foo.bar* is identified via a DNS mapping of type NS, from the domain name to the domain name of the name server, which *could* be subdomain, e.g,. *ns1.foo.bar*, or not, e.g., *ns.goo.net*. Mappings of a domain name, e.g., *x.foo.bar*, are trusted only if received from a name server of that domain or of a parent domain, e.g., the name server of *foo.bar* or of *bar*.

Clients use *resolvers* in order to find RRs for a domain. The resolvers query the name servers to locate the requested RRs. Upon query, a name server responds with the corresponding RR, or a *non-existing domain* response in case no matching RR exists. Resolvers cache the DNS responses; the caching time is specified in the time to live (TTL) field of a response, e.g., TTL of t seconds indicates that the resolver should store the record for t seconds. Subsequent requests for the same RRs are provided from the cache. A sample lookup process, initiated with a DNS request from a stub resolver, is depicted in Figure 2.

2.2 DNS cache-poisoning

In this section, we provide background on DNS cache poisoning.

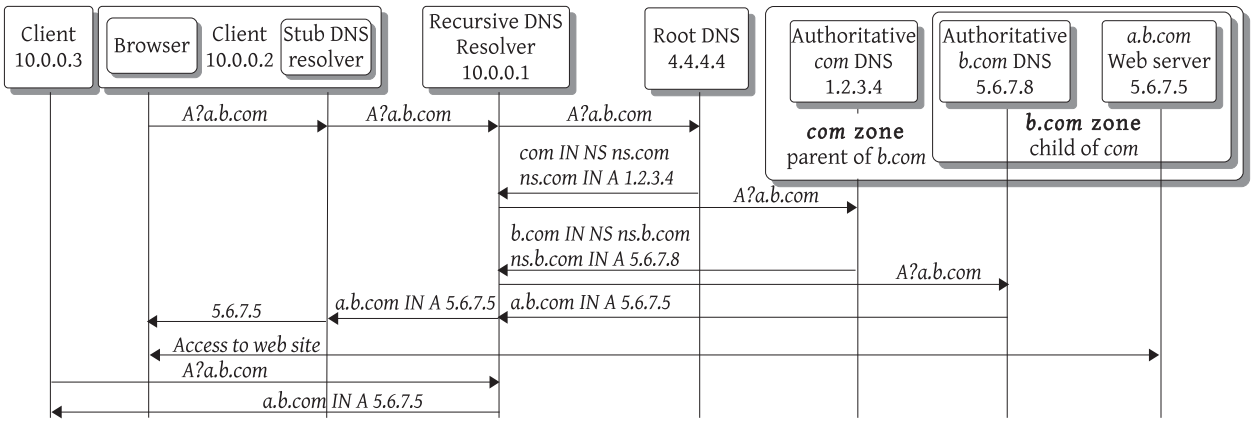

Figure 2 Sample resolution process initiated by a stub resolver for an IP address of a web site. Recursive resolver performs the lookup and caches the resource records from the DNS response. Subsequent requests from clients for the same RRs are satisfied from the cache.

The resolvers accept only responses for which there are corresponding pending DNS requests; thus, the first step in a DNS cache poisoning attack is to trigger a DNS request. Then, after triggering the request, the second step is to inject a spoofed response (redirecting the clients to incorrect hosts) that will be accepted and cached by a victim resolver.

The attackers can use a number of techniques for triggering DNS requests; some techniques attack random victims while others are targeted against specific users.

When an attacker has a direct access to a victim DNS resolver it can repeat the cache poisoning attack and trigger arbitrary number of requests at will. This is possible if the attacker is on the same network with the victim resolver, e.g., it is one of the clients of an ISP whose resolver it wishes to poison.

Another option is to use an open DNS resolver. The amount of open resolvers on the Internet is constantly increasing, from 15 million in 2010 [18] to 30 million in 2013 [19]. Open resolvers provide recursive DNS services to any requesting client. This ability to trigger DNS requests makes open resolvers more vulnerable to attacks and, in particular, to DNS cache poisoning attacks.

Typically, DNS resolvers limit their recursive DNS service only to clients on their networks. However, there are techniques which the attackers can use to trigger requests even remotely.

A known technique is by controlling a malicious script, typically dubbed 'puppet' (sandboxed client) [20], such as a client running Javascript or presenting Flash content. The attacker can accomplish this, for instance, by purchasing an ad space from advertising web site. When clients surf to such web sites, their browsers are redirected, e.g., via images, or iframes, to retrieve objects from other domains (in this case from attacker's domain). This causes the browser to load the resource from a different (remote) domain. Once redirected, the browsers of the clients download and run the script and become puppets. The script runs automatically and without any interaction with the client; see Figure 3. The script can trigger DNS requests to domains and responses to which the (external) attacker wishes to poison.

An attacker can also trigger DNS requests via other, less known techniques, e.g., by sending email to a victim network whose resolver it wishes to poison, see Figure 4; this technique was first proposed in [21].

3 DNS cache poisoning by man-in-the-middle

Basic Internet protocols, such as DNS and routing, are not cryptographically protected against MitM adversaries, and current defences rely on challenge-response mechanisms which provide security only against off-path adversaries. Deploying cryptography is more difficult than merely configuring challenge-response mechanisms, and the *incorrect* belief is that since MitM is not common, such defences should suffice.

In this section, we argue that this belief is wrong and review a number of common scenarios where adversaries possess MitM capabilities. We show how easily the adversaries can exploit that ability and how devastating the results may be for the victims.

3.1 MitM in open access networks

Since the early days of the Internet and until recently, clients have accessed the Internet mostly from trusted networks, e.g., via their ISPs or enterprise networks. However, during the last decade, an increasing number of devices obtain Internet connectivity via public IEEE 802.11-based wireless (Wi-Fi) networks [RFC5416], e.g., hotels, airports, cafes or networks set up by individuals.

The threats of using such open networks are twofold and stem from a (1) malicious operator and a (2) malicious client.

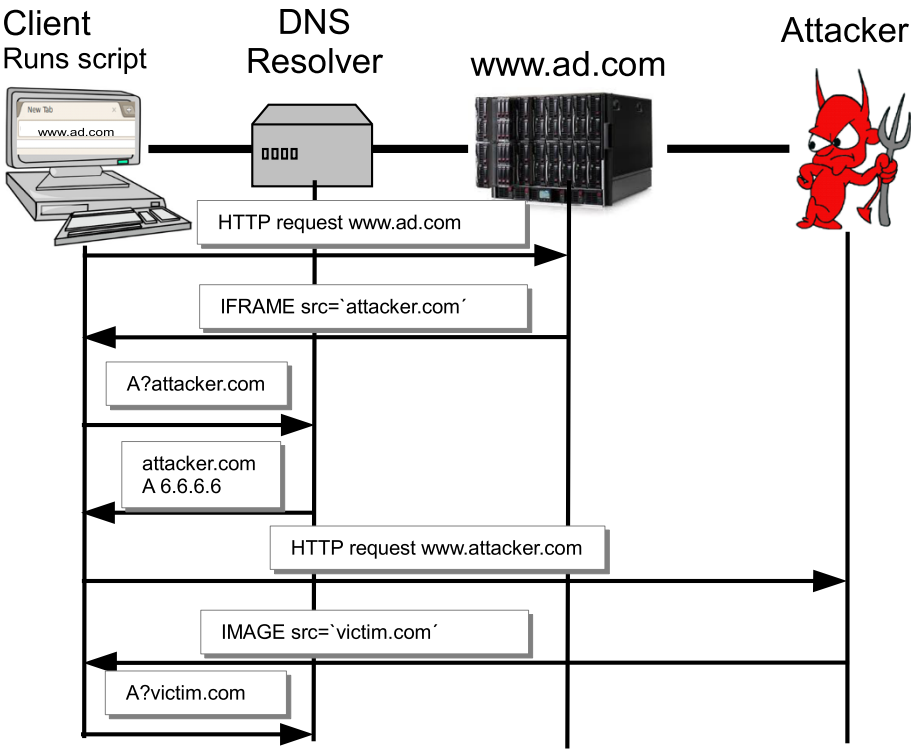

Figure 3 Triggering DNS requests via a malicious script (puppet).

Both, a malicious operator and a malicious client, can subvert correctness and availability of Internet services for their clients. The most effective attack is by spoofing DNS responses and redirecting the clients to incorrect (malicious) hosts, e.g., to download malware or block responses to launch a denial/degradation of service attacks. A malicious operator essentially has MitM capabilities on its network since the traffic of all the clients connected via its network traverse a router under its control. In particular, the network operator can block correct responses and craft spoofed responses from scratch or can inject spoofed records into DNS response packets.

A malicious client can gain MitM capabilities by spoofing DHCP responses for newly connecting clients. In spoofed DHCP responses, a malicious client can provide an incorrect IP/MAC address for local recursive DNS resolver, e.g., one that is assigned to its own network interface card (NIC), and thus will receive all the DNS requests sent by the victim client. But, MitM capabilities are not essential for launching attacks on public wireless networks. In particular, every connected device can receive all transmissions, no matter who the destination is. This enables malicious clients to inspect all DNS requests, i.g., packets sent to port 53, and to craft spoofed responses before authentic responses arrive.

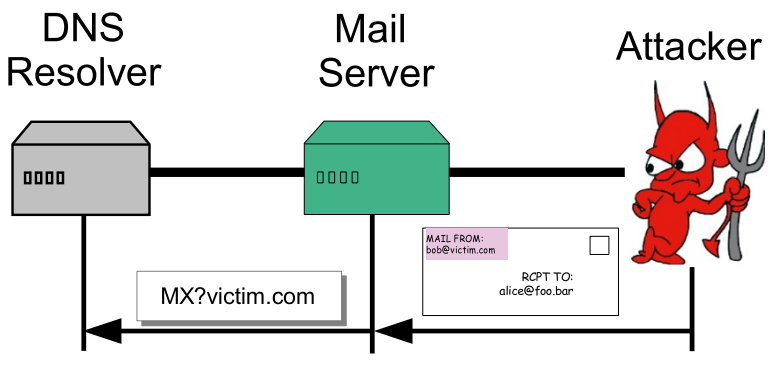

Figure 4 Triggering DNS request by sending email to a victim network (simplified).

However, attacks on correctness and availability are not surprising and both are known threats. In what follows, we outline a less evident threat which appears to be gaining relevance during the recent couple of years. Specifically, we are referring to monitoring online user activities. The network operator, as well as other clients, can inspect the MAC address of all the clients connected to that network. MAC address (uniquely) identifies a network adapter and has the same value no matter which network the client connects to the Internet from. This enables tracking the users throughout the different networks that they use to connect to the Internet. Even benign network operators may pose a threat, e.g., the logs may be kept over a long time period, and may be shared with third parties.

Such logs enable different parties, e.g., security agencies, armies, content providers, to learn about the online behaviour and habits of the clients.

3.2 MitM on backbone links

Recent revelations [10] expose monitoring and censorship activities of National Security Agency (NSA) and Government Communications Headquarters (GCHQ) against Internet services and users. The NSA used its secret agreements with telecommunications companies to monitor communications channels in order obtain access to collect and analyse Internet traffic. The NSA uses hosts (code name QUANTUM) to inject spoofed DNS (and HTTP) responses: when observing a DNS request, a spoofed response is automatically crafted and returned to the victim client. Notice that since the QUANTUM servers are deployed on the backbone links, they can always respond before the legitimate server does. Since the first correct response is accepted and the subsequent ones are ignored, the attacker can redirect victim clients to the servers controlled by the NSA (code name FOXACID); the servers then install malware on clients hosts or tap on the communication.

3.3 MitM via route poisoning

The Internet consists of multiple autonomous systems (ASes) which are interconnected by means of routing protocols. To enable connectivity, the networks advertise their prefixes, i.e., address blocks, to the Internet via a BGP update messages, [RFC1771]. Every BGP update message is an indication of a routing change. Routers issue BGP update messages when routing information changes, e.g., link failures, topology changes, reconfigurations, updates of local policies. A BGP update contains an advertised prefix and an AS path. The last AS on the path is the originator of the prefix. Since BGP does not employ authentication mechanisms, originators of BGP routing announcements may claim prefixes belonging to other networks or may change routing path (by adding or removing links), e.g., due to benign failures or malicious attacks. Attackers can hijack prefixes by advertising invalid origin or invalid next hop [14]. There is a large body of research studying attacks on BGP routing, e.g., route hijacking and route injection that damage network operation or connectivity.

For instance, recently, a highly publicised route hijacking attack was exposed [15], which was launched over a period of a number of months, and the attackers routed a significant amount of traffic through Belarus and Iceland. Belarus Telecom was advertising a false route and thus managed to hijack traffic which was not directed to its prefix. Such route hijacks were believed to be theoretical prior to that attack, and it showed the feasibility of such massive MitM hijacking.

Route poisoning can be employed to leverage DNS cache poisoning attacks. By forcing the traffic to traverse a specific path, e.g., via a malicious network operator, the attacker can become a MitM for the communication to a target domain and can easily inject spoofed DNS responses into the traffic flow.

4 Cache poisoning by subverting hosting infrastructure

Many DNS cache poisoning attacks occur by subverting the hosting infrastructure of DNS, e.g., domain registrar or name servers. Indeed, there is an increasing number of attacks by compromising the hosting side of DNS which allows to take over victim domains. Subverting a registrar or a name server is a lucrative avenue for cache poisoning when the attacker is not a MitM.

Attacks compromising the hosting infrastrucutre are frequently occurring. In 2013 alone, multiple domains were hijacked by compromising domain name servers or registrars; some of the notable attacks include a compromise of `google.rw` and even top level domains like, `qa, ps, nl, be, my`. Registrar, `register.com` was subverted, and as a result, many names related to security like `metasploit.com` and `bitdefender.com` were redirected.

4.1 Compromising registrars

Attackers can exploit vulnerabilities, most notably, in the user interface, provided by the registrars. In particular, attackers often exploit vulnerabilities in user interface, such as lack of (or insufficient) user input validation to perform injection attacks, e.g., buffer overflow, and obtain a shell on the victim host. This allows to manipulate DNS records in the zone file, resulting in cache poisoning attacks of the target domain. This serves as a stepping stone to multiple attacks, e.g., enables attackers to distribute spam while passing reputation-based spam filters, to perform phishing, malware distribution, credentials theft.

For instance, a security hole within 123 REGISTRATION registrar management console resulted in the hijacking of 300 domains back in 2012; the problem was eventually tracked down to an open account control panel that had allowed changes to be made without adequate authentication.

We tested the interfaces of a number of popular registrars and found a vulnerability which may facilitate cache poisoning attacks, even *without* compromising the web interfaces: when registering a domain, the attackers can configure legitimate name servers that belong to other domains and are not under their control as their own.

This fact can be abused, e.g., for cache poisoining or denial of service attacks. For instance, consider an attacker that registers a domain under some top level domain, e.g., `one-domain-to-rule-them-all.org`, and registers a name server that belongs to another domains under `org`. Then, referral responses to requests for resource records within attacker's domain can be exploited to poison the records of the victim domain whose name server record the attacker used.

Unfortunately, detecting and preventing such attacks is challenging and would require the registries to validate the ownership over the records at registration time.

4.2 Compromising name servers

There is a long history of attacks exploiting vulnerabilities in the name servers, e.g., vulnerable operating system or vulnerable DNS software. This is a stepping stone to obtain unauthorised access to the system and to execute arbitrary code. Vulnerabilities were registered in popular operating systems and DNS software, e.g., MS server, Bind versions, PowerDNS.

Some of the known attacks exploited known vulnerabilities, such as (1) buffer overflow, which allows an attacker to obtain unauthorised access to the system and execution of arbitrary code, (2) inproper handling of input values, e.g., one attack exploited vulnerable error handling routine that would crash on invalid DNS transaction identifier values, (3) improper check of memory copy, e.g., would crashed the server allowing an attacker to gain root privileges on name server, and many others.

5 Dependencies in Domain Name System

Dependencies within the DNS infrastructure further exacerbate the impact of cache poisoning attacks or compromises of name servers or registrars [11]. We consider the following types of dependencies: (1) inter-domain dependencies via transitive trust and (2) zones coresidence due to name servers sharing and (3) dependencies via registrars.

Our study encompassed top 50 K Alexa domains and 568 TLDs. We also measured all the domains depending on these Alexa domains and TLDs via a transitive trust, which resulted in a total of 150 K domains. These domains are served by 48 K name servers; these 48 K name servers have 65 K different IP addresses, since sometimes a single name server is assigned a number of IP addresses.

5.1 Transitive trust

A transitive trust dependency can be twofold: (1) a name server can appear in a number of transitive trust chains (i.e., the number of domains that can be impacted by a failure of a specific name server) and (2) a domain can depend on multiple domains for its resolution (i.e., impact of a failure of a single server on the latency or availability of a domain). The former impacts the resilience of the DNS infrastructure and the latter the resilience of a specific domain. Ideally, both should be low.

Our study shows that on average, a domain in top 50 K Alexa depends on 43.5 other domains via transitive trust chains, and on average, a domain in TLD depends on 43.7 domains via transitive trust chains. The maximal number of transitive trust dependencies in Alexa domains is 220 and in TLDs is 183. For instance, domain `sigcomm.com`, ranked 373097 on Alexa, is hosted at `dnsmadeeasy.com`, coresiding with 400 other domains.

Figure 5 plots the cumulative distribution function (CDF) $F(x) = \Pr[X \leq x]$ of the *number of name servers appearing in transitive trust chains* of top 50 K Alexa domains and TLDs; this encompasses dependency (1) - and increases traffic volume to name servers and resolution latency for clients. The CDF curves for the Alexa domains and TLDs represents the number of transitive trust chains that name servers in Alexa domains (resp. TLDs) appears in. Approximately, 50% of name servers appear in two or more transitive trust chains. More than 90% of the name servers appear in eight and less chains. Some name servers appear in more than 128 transitive trust chains.

Figure 6 plots the CDF of the transitive trust dependencies of Alexa top 50 K domains and TLDs; this expresses dependency (2) - and increases resolution time for clients'. Approximately, 50% depend on 20 or more other domains for their resolution, and more than 90% depend on more than 128 domains.

Name servers with high dependencies via transitive trust, i.e., whereby multiple domains depend on them for their resolution, have two side effects: (1) they can become a lucrative target for attacks. For instance, recently, Herzberg and Shulman [3] showed how to launch a DNS cache poisoning attack using fragmented DNS responses to replace the authentic IP address of a victim name server with a spoofed one and ran this against `sns-pb.isc.org` name server, which appears in 69 transitive trust chains of other domains.

(2) Another notable side effect is that large transitive trust chains introduce more latency to resolution

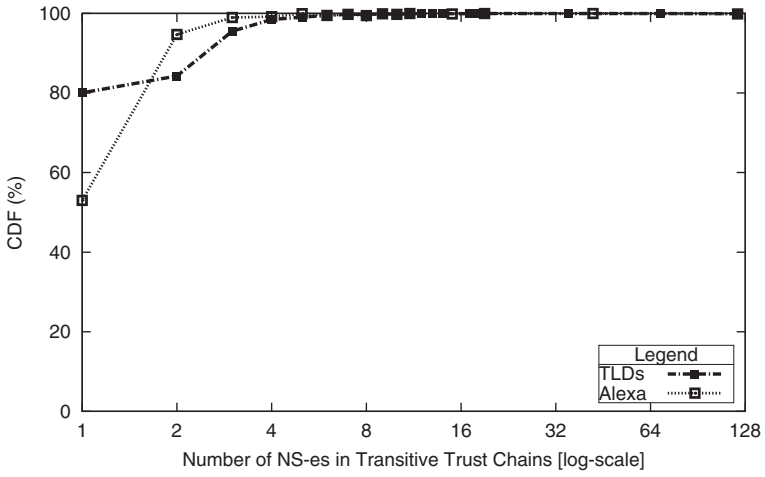

Figure 5 Name servers appearing in transitive trust chains of top 50 K Alexa domains and TLDs.

of records within domains depending on many other domains and increase the queries rate to name servers appearing in multiple transitive trust chains. Our study measured an increase of 50 ms for every transitive trust chain of three links, when measured with a cold (empty) cache. Resolutions of larger chains, e.g., 200, can often result in timeouts and unnecessary retransmissions, overloading the network and the name server, and increasing the latency for clients' queries.

Transitive trust dependencies also nullify effectiveness of DNSSEC, [RFC4033-RFC4035], and impede its adoption. In particular, if name servers or other resources of a signed zone are placed under unsigned domains, the DNS resolver will not be able to establish the security of the signed records, and the security will depend on the security of the weakest link in a transitive trust chain.

5.2 Coresidence

Hosting a number of zones files on the same name server enables DNS name server operators to optimise profit and reduce operational costs and management overhead. We measure and quantify the dependencies between zones, namely *the fraction of zone files residing on the same physical server*. We measure the coresidence among TLDs and top 50 K Alexa domains, including the coresidence between name servers appearing in their transitive trust dependencies. As our results, plotted in see Figure 7, indicate, the coresidence rate among the name servers are extremely high. We found that coresidence of multiple zones on the same name server is a common practice among Alexa domains and TLDs. In particular, more than 70% of the name servers of Alexa domains and more than 80% of the name servers of TLDs host multiple zones.

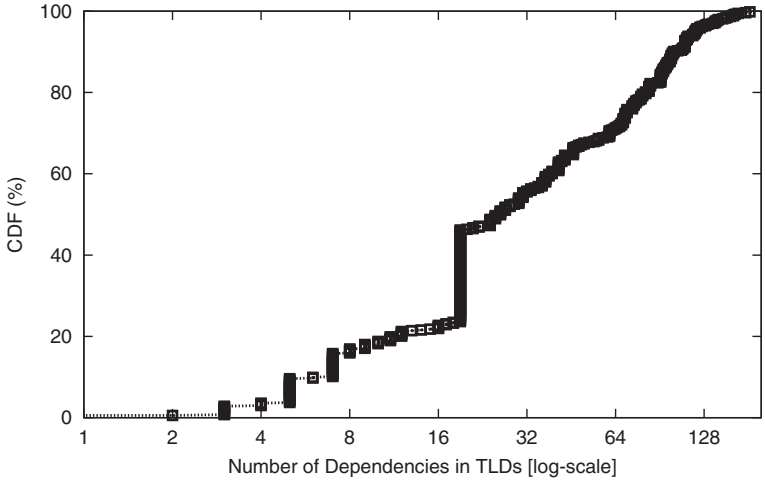

Figure 6 Transitive trust dependencies in top 50 K Alexa and TLDs.

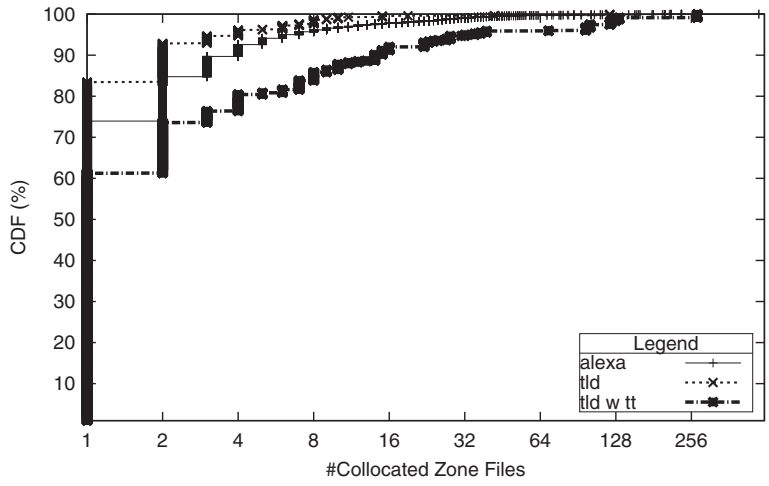

Figure 7 Zones coresidence on name servers among 50 K-top Alexa domains and TLD and servers in transitive trust dependencies of Alexa domains and TLDs.

Some name servers host more than 500 zone files, such as the name server `pdns.ultradns.net`.

The implication of high coresidence rate is that a failure or a DoS attack against the availability of the name server or the network hosting it impacts the availability of *all* of the zones hosted on it. An attack against a name server, e.g., exploiting a vulnerability in a DNS software or in the operating system can enable the attacker to take control over the name server and inject records into the zone files hosted on it. We also find that high coresidence increases loss and failures.

5.3 Dependencies via registrars

Many DNS cache poisoning attacks occur by subverting the hosting infrastructure of DNS, e.g., domain registrar or name servers. By exploiting social engineering or vulnerabilities in domains registration interface, an attack against an infrastructure, run by a registrar, may expose all of the domains using it to domain hijacking attacks. Subverting a registrar or a name server is a lucrative avenue for cache poisoning when the attacker is not a MitM. Indeed, there is an increasing number of attacks by compromising the hosting side of DNS.

We quantify the dependencies between domains and registrars. In our study, we measured dependencies between 1003 ICANN accredited registrars and 18 popular TLDs. Figures 8 and 9 depict the inter-dependencies between registrars and domains. In particular, Figure 8 shows dependencies of popular TLDs. On average, a domain depends on almost 600 registrars, with the maximum reaching over 1 K as in the case of the `com` TLD. Attack against any of these registrars, on which a single

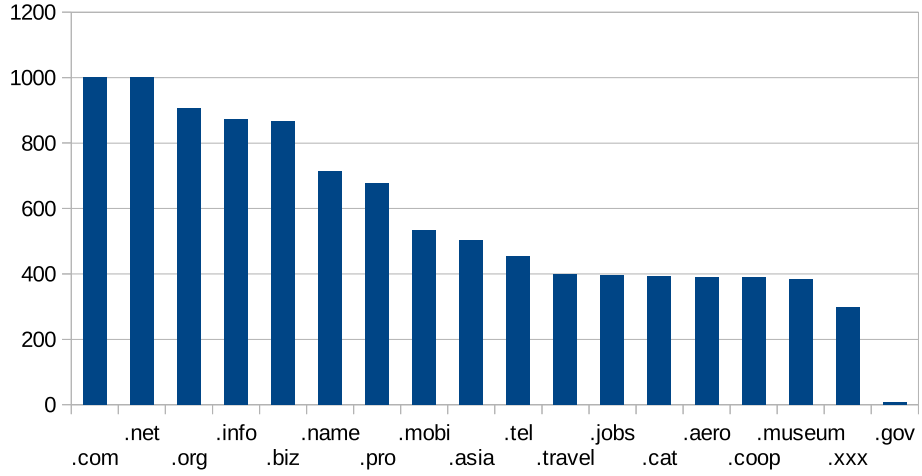

Figure 8 Dependencies between TLDs and registrars accredited to manage them.

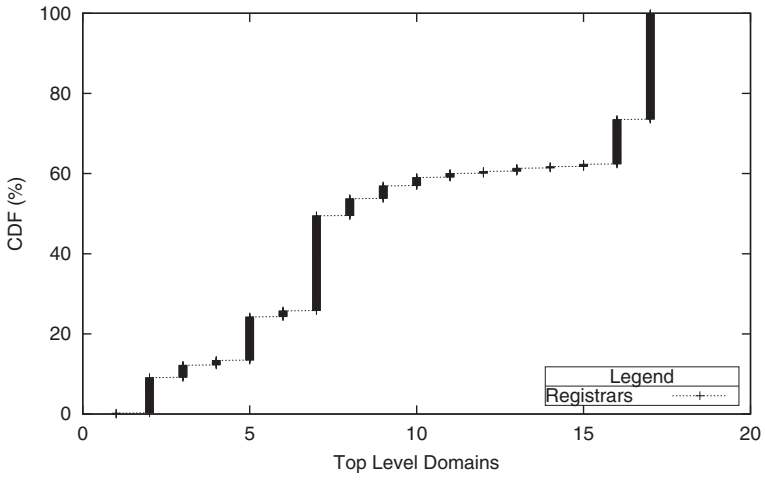

Figure 9 Dependencies between TLDs and registerars, showing fraction of TLDs that registrars are accredited to manage.

domain depends, may suffice to hijack its subdomains and even the domain itself. Figure 9 shows the fraction of TLDs under which registrars can register subdomains. The implication is that subverting a single registrar may allow the attacker to hijack or register subdomains under all the TLDs that the registrar is accredited to manage.

6 DNS security extensions

Due to the critical function that DNS fulfills, it is highly reactive to new threats and attacks, constantly responding with new defence mechanisms. In particular, to mitigate the detrimental damages of cache poisoning attacks, IETF designed and standardised a cryptographic defence for DNSSEC [RFC4033-RFC4035]. DNSSEC 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.

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`, [22]), secure email, (with PGP keys distribution [23]). For the protection of DNSSEC to kick in, the zones need to be signed starting with the *trust anchor* (the root zone) all the way down to the target domain to enable the resolver to establish a chain of trust. The name server has to serve signed responses, and the resolver has to validate the signatures and keys in the responses; see an illustration of the chain of trust establishment process in Figure 10.

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 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

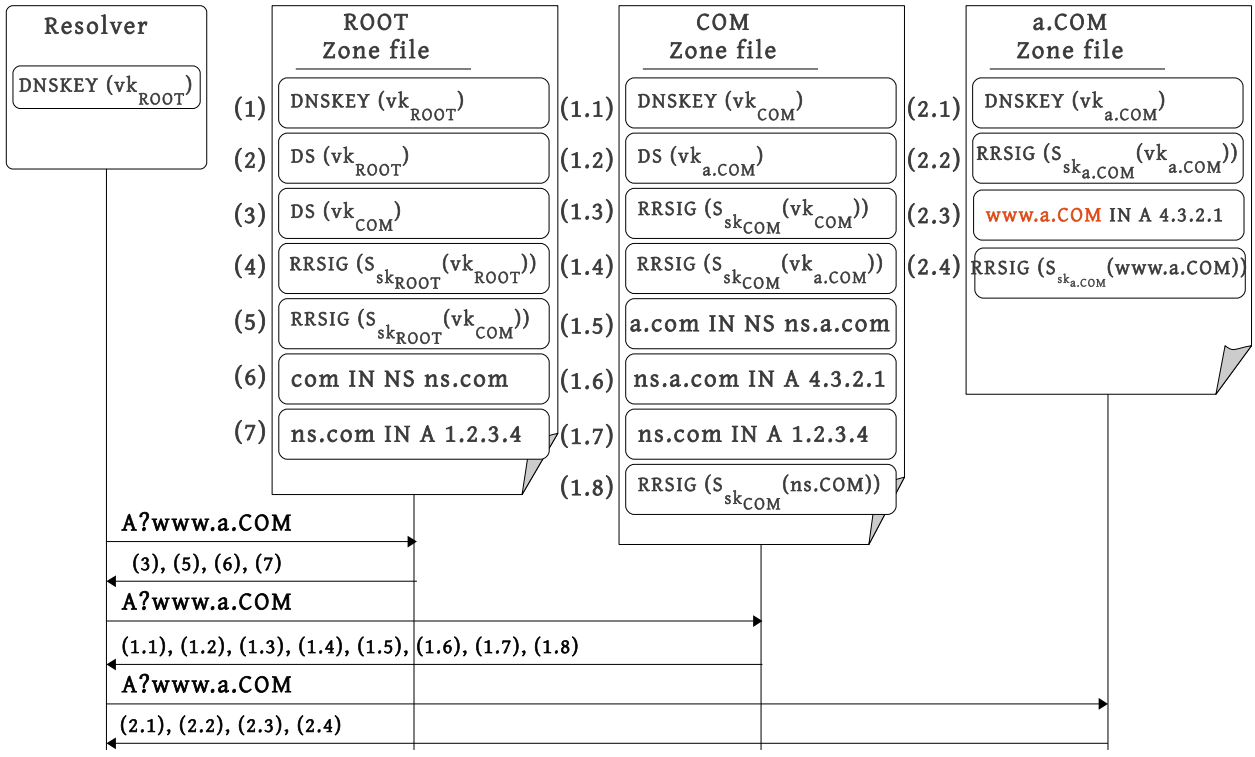

Figure 10 A (simplified) sample process of constructing a chain of trust from the root zone. For ease of presentation in this illustration, the RRs maintained by the name servers are enumerated, and we specify the exchanged RRs by indicating the corresponding numbers above the arrows.

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.

6.1 DNSSEC deployment

Although proposed in 1997, DNSSEC is still not widely deployed: less than 3% of the DNS resolvers validate DNSSEC records in DNS responses [24]; recent measurements of DNSSEC adoption on forward and reverse DNS trees show that less than 1% of the zones are signed, [25].

We summarise the results in Table 1. The first column contains the number of registered domains that were tested. This is mainly relevant in the reverse DNS tree, where a large fraction of the domains (that correspond to IPv4 address blocks) are not registered.

Table 1 DNSSEC deployment measurements in forward and reverse DNS trees

Domain	Registered	Name servers	Signed
Forward DNS Alexa	50 K	32.5 K	0.46%
Forward DNS TLDs	568	3.2 K	62%
rDNS `x.in-addr.arpa.`	229	1.5 K	84%
rDNS `y.x.in-addr.arpa.`	28 *K*	97 K	0.4%
rDNS `z.y.x.in-addr.arpa.`	2,767 *K*	9,687 K	0.06%

The second column contains the number of name servers in each domain space. In forward DNS, 62% of the TLDs are signed and less than 1% (0.46%) of top million Alexa domains are signed.

In a reverse DNS, there are only 0.07% signed zones in total; this is surprising since the reverse DNS is commonly utilised by security mechanisms. Notice that very few of the reverse DNS domains, that correspond to classes B and C (i.e., `x.y.in-addr.arpa` and `x.y.z.in-addr.arpa`), are signed (only 0.4% and 0.06% respectively). This means that even if the lower domains decide to adopt DNSSEC, the resolvers will not be able to establish a chain of trust to them.

Such a low adoption of DNSSEC among domains may be an indication of deployment challenges that needs to be investigated and mitigated. We hope that our work will further motivate such study.

6.2 Operational challenges

There are a number of challenges related to deployment of DNSSEC, which we studied in our earlier work [26,27]. In this section, we discuss *operational* challenges and *outages*. According to our study, the outages are mainly related to errors in key rollover and to zone

signingprocedure. For instance, in January 2012, Comcast (a large ISP) stopped serving responses for `nasa.gov`. This immediately incited speculations whether Comcast was blocking `nasa.gov`. In reality, `nasa.gov` served incorrect signatures over its DNS records, and the validating resolvers of Comcast discarded those 'invalid' responses; the resolvers of Comcast were functioning correctly since such incorrectly signed records could also constitute an attack.

In August 2013, a mistake in key rollover, whereby instead of signing with both the old key and the new one, only the new key was used, causes an outage of domains under `gov` TLD. The impact was that 18 million clients of comcast Internet provider, more than 70 customers of Google Public DNS and validating Internet providers all over the globe (e.g., Sweden, Czech, Brazil) could not access domains under `gov` TLD.

There were also a number of other publicised failures, which resulted in broken DNS functionality for victim networks. Most, if not all, of the failures are related to human errors, and operational challenges in DNSSEC could be mitigated by automating the signing and key rollover procedures.

6.3 Forensics, evidences and detection with DNSSEC

In this section, we show that DNSSEC can be useful for detection of attacks and in forensic analysis. The feature that makes this possible are digital signatures, which can be validated and verified by anyone with the possession of a public verification key. Signatures provide a valuable information for forensic analysis and can enable identification, e.g., of the exact time that the network was attacked and to which hosts the traffic was redirected.

In contrast to the relative time indicated in the TTL field in DNS records, the cryptographic signatures contain an absolute expiration date and the date the signature was generated. The signatures also contain the tag of the cryptographic material (algorithm, hash, key) that was used to produce the signature.

Notice that since the cryptographic signatures should be impossible to forge for efficient adversaries, only the entity in possession of a cryptographic signing key, or a very strong adversary with huge computational power, should be able to craft valid signatures. Thus, a forged signature is an indication of a very strong adversary, such as a government, or an indication that the zone, which provided the spoofed record with a valid signature, may be malicious or subverted. In what follows, we consider how to analyse attacks or detect breaches, performed by strong adversaries, *a posteriori*.

We propose to utilise DNSSEC to design a system that would enable analysis of attacks, provide evidences of attacks that took place and even enable detection of some attacks. The system would need to collect the DNS responses (along with the corresponding signatures and cryptographic keys), e.g., by configuring suitable rules in the firewall, and store them in a database for processing.

6.3.1 Forensic analysis

The time stamps on the signatures provide a valuable information, allowing to analyse when a certain mapping was considered valid, and when it constitutes an attack. For instance, consider an organisation that had a network block 1.2.3.0/24 and then moved to a different Internet provider and received a new address block 5.6.7.0/24. All the servers that once occupied a block 1.2.3.0/24 were also moved to address block 5.6.7.0/24. Thus, responses with mappings from the block 1.2.3.0/24 are no longer valid, and if a resolver on some network receives records with mapping from old block, this may be an indication of an attack.

The time fields in the signatures (over the DNS records) enable network operators to analyse when the spoofed records were supplied and if the records reflect the real mappings at the time that they were supplied.

6.3.2 Evidences

It may often be desirable to prove to a third party, e.g., judge, registrar or domain operator, that attack took place. For instance, consider a case where a customer's private data was breached via a redirection to a malicious host. The customer can present the malicious records (which were used in the course of the attack) along with the cryptographic signatures, to a third party, and any third party can be convinced by validating the signatures. Another example is of a stronger adversary, for instance a state, that forces `com` domain to redirect all traffic destined to one of its subdomains, e.g., a Chinese enterprise Huawei `Huawei.com`, to different servers. Since `com` signs the delegation records for `Huawei.com`, it can also produce valid signatures for those new servers. If the attack is detected, those signatures can indicate that the incident was not a benign failure or mistake, but a malicious attack, which involved resigning the delegation records belonging to `Huawei.com`.

Such evidences are not available with other cryptographic defences that were proposed for DNS, most notably Eastlake cookies [16] and DNSCurve [17].

6.3.3 Detection

DNS is a distributed infrastruture, and a single domain is often served by mutliple name servers. Furthermore, many name servers are also distributed, e.g., via the ANYCAST technology, [RFC1546], where a DNS request is rerouted to the topologically nearest name server.

The attacks that we discussed in Sections 3 and 4 were launched against a specific name server or a traffic that was exchanged with a specific instance of the name server, e.g., the attacker either subverted a name server, or injected spoofed responses into a communication flow with a name server. Indeed, it is much more difficult (if not impossible) to subvert *all* of the name servers of some target domain. This would require an attacker that can eavesdrop on multiple Internet links that belong to different autonomous systems (AS) or to compromise all the name servers. Since this should be impossible even for states and military organisations, we use this as a basic premise in the detection technique which we propose next.

The fact that the adversary can compromise only some of the links and servers means that the different name servers (and different instances thereof) will return different responses to DNS requests.

Networks could establish trustworthiness and correctness of the DNS responses, by querying the different name servers (and instances thereof), belonging to target domains. To query name servers instances distributed via an ANYCAST, the network could use proxies located in different parts of the Internet. The inconsistencies, if found, would be carefully checked to test for attacks.

7 Conclusions

A secure DNS is critical for stability and functionality of the Internet. In this work, we review DNS security. We show that current defences do not suffice against common attackers and the DNS infrastructure is vulnerable to cache poisoning. We also experimentally measure the dependencies within DNS. Our results indicate that attacks against a single weak link can impact multiple dependent domains.

A significant problem pertaining to many attacks is that it is impossible to detect them (unless they break connectivity or disrupt the 'expected' functionality). We show that DNSSEC could not only prevent many of the attacks but could also be used to enable detection of the attacks and as well as *a posteriori* forensic analysis of the attacks. DNSSEC also can be used to generate cryptographic evidences, which would enable victims to prove to the third parties, e.g., insurance organisations, judges, Internet operators, that attacks and breaches took place. To best of our knowledge, our work is the first to propose such applications of DNSSEC.

However, deployment and operation of DNSSEC are challenging; we discuss problems and recommend countermeasures. We hope that our work will raise awareness to the vulnerabilities and the potential of DNSSEC to facilitate forensic analysis of attacks.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This research was supported by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE, and by the Hessian LOEWE excellence initiative within CASED.

Author details
[1]Fachbereich Informatik Technische Universität Darmstadt Mornewegstraße 32, Darmstadt DE, Germany. [2]Fraunhofer Institute for Secure Information Technology SIT Rheinstraße 75, Darmstadt DE, Germany.

Received: 13 August 2014 Accepted: 26 September 2014
Published online: 22 October 2014

References

1. A Herzberg, H Shulman, in *ESORICS. LNCS*, ed. by S Foresti, M Yung, and F Martinelli. Security of patched DNS, vol. 7459 (Springer Pisa, Italy, 2012), pp. 271–288. http://dx.doi.org/10.1007/978-3-642-33167-1
2. A Herzberg, H Shulman, in *European Symposium on Research in Computer Security (ESORICS). Lecture Notes in Computer Science*. Vulnerable Delegation of DNS Resolution (Springer Egham, UK, 2013)
3. A Herzberg, H Shulman, in *The Conference on Communications and Network Security, CNS 2013 IEEE*. Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org (IEEE Washington, D.C., U.S, 2013)
4. A Herzberg, H Shulman, in *ACM Annual Computer Security Applications Conference (ACM ACSAC)*. Socket Overloading for Fun and Cache Poisoning (New Orleans, Louisiana, U.S., 2013)
5. H Shulman, M Waidner, in *Applied Cryptography and Network Security (ACNS)*. Fragmentation Considered Leaking: Port Inference for DNS Poisoning (Springer Lausanne, Switzerland, 2014)
6. H Shulman, The (in) Security of Outsourced DNS. Technical Report (August 2014)
7. P Neira, Patchset of Netfilter Updates (2013). http://patchwork.ozlabs.org/patch/307041/
8. HF Sowa, ipv4: introduce new IP MTU DISCOVER mode IP PMTUDISC INTERFACE. Linux Kernel Mailing List (2013). http://thread.gmane.org/gmane.linux.network/288482
9. National Security Agency, PRISM (surveillance program) (2007)
10. National Security Agency, FOXACID and QUANTUM Programs (2013)
11. H Shulman, S Ezra, in *ACM Conference on Computer and Communications Security (CCS)*. Poster: On the Resilience of DNS Infrastructure (ACM Scottsdale, Arizona, U.S., 2014)
12. V Ramasubramanian, E Sirer, in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. Perils of transitive trust in the domain name system (USENIX Association, 2005), pp. 35–35
13. F Denis, The GOOGLE.RW Hijack (2013). http://labs.umbrella.com/2013/10/25/google-rw-hijack-nobody-else-noticed/
14. H Ballani, P Francis, X Zhang, in *ACM SIGCOMM Computer Communication Review*. A Study of Prefix Hijacking and Interception in the Internet, vol. 37 (ACM, 2007), pp. 265–276
15. K Bode, Somebody is Hijacking Massive Amounts of Data Via Iceland (2013). http://www.dslreports.com/shownews
16. D Eastlake, Domain Name System (DNS) Cookies (2014). https://tools.ietf.org/html/draft-eastlake-dnsext-cookies-04
17. DJ Bernstein, DNSCurve: Usable security for DNS (2010). Posted at: http://dnscurve.org/
18. D Leonard, D Loguinov, in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. Demystifying service discovery: implementing an internet-wide scanner (ACM, 2010), pp. 109–122
19. Open DNS Resolver Project, Last retrieved May 2013. http://openresolverproject.org/
20. S Antonatos, P Akritidis, VT Lam, KG Anagnostakis, Puppetnets: Misusing web rowsers as a distributed attack infrastructure. ACM Trans. Inf. Syst. Secur. **12**(2), 12–11215 (2008)
21. A Herzberg, DNS-based email sender authentication mechanisms: a critical review. Comput. Secur. **28**(8), 731–742 (2009)
22. J Gersch, D Massey, in *Computer Communications and Networks (ICCCN) 2013 22nd International Conference On*. Rover: Route origin verification using dns (IEEE, 2013), pp. 1–9

23. P Wouters, Using DANE to Associate OpenPGP public keys with email addresses (2014). http://tools.ietf.org/html/draft-wouters-dane-openpgp-02
24. W Lian, E Rescorla, H Shacham, S Savage, in *Proceedings of USENIX Security*. Measuring the Practical Impact of DNSSEC Deployment, (2013)
25. H Shulman, Is DNSSEC Ready for Prime Time: The Challenge of Legacy Infrastructure. Technical Report (September 2014)
26. A Herzberg, H Shulman, in *Eighth International Conference On Availability, Reliability and Security (ARES), 2013*. Dnssec: Interoperability challenges and transition mechanisms (IEEE Regensburg, Germany, 2013), pp. 398–405
27. A Herzberg, H Shulman, in *Communications and Network Security (CNS) 2013 IEEE Conference On*. Dnssec: Security and availability challenges (IEEE, 2013), pp. 365–366

doi:10.1186/s13635-014-0016-2
Cite this article as: Shulman and Waidner: DNSSEC for cyber forensics. *EURASIP Journal on Information Security* 2014 **2014**:16.