2014 IEEE Security and Privacy Workshops

Towards Forensic Analysis of Attacks with DNSSEC

Haya Shulman and Michael Waidner
Fachbereich Informatik
Technische Universität Darmstadt
{haya.shulman, michael.waidner}@cased.de

***Abstract*—DNS cache poisoning is a stepping stone towards advanced (cyber) attacks, and can be used to monitor users' activities, for censorship, to distribute malware and spam, and even to subvert correctness and availability of Internet networks and services.**

The DNS infrastructure relies on challenge-response defences, which are deemed effective for thwarting attacks by (the common) off-path adversaries. Such defences do not suffice against stronger adversaries, e.g., man-in-the-middle (MitM). However, there seems to be little willingness to adopt systematic, cryptographic mechanisms, since stronger adversaries are not believed to be common.

In this work we validate this assumption and show that it is imprecise. In particular, we demonstrate that: (1) attackers can frequently obtain MitM capabilities, and (2) even weaker attackers can subvert DNS security. Indeed, as we show, despite wide adoption of challenge-response defences, cache-poisoning attacks against DNS infrastructure are highly prevalent.

We evaluate security of domain registrars and name servers, experimentally, and find vulnerabilities, which expose DNS infrastructure to cache poisoning.

We review DNSSEC, the defence against DNS cache poisoning, and argue that, not only it is the most suitable mechanism for preventing cache poisoning attacks, but it is also the only proposed defence that enables a-posteriori forensic analysis of attacks. Specifically, DNSSEC provides cryptographic evidences, which can be presented to, and validated by, any third party and can be used in investigations and for detection of attacks even long after the attack took place.

***Keywords*—security; cyber attacks; DNS cache-poisoning; DNSSEC; digital signatures; cryptographic evidences;**

I. 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 organisations and exploit vulnerabilities in Internet systems and services, e.g., WEB and cloud, as well as in basic building blocks of the Internet, such as Domain Name System (DNS) and routing. The attacks inflict economical losses to businesses and have a devastating impact on ecommerce, security, and critical infrastructures.

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, can be applied for phishing attacks, credentials theft, eavesdropping.

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.

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 forged 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], [2], [3], [4], [5]. Some of these vulnerabilities were already patched, e.g., [6], [7]. These attacks in tandem with the recent revelations on the surveillance programs by the National Security Agency (NSA), [8], [9], raise the question whether the widely deployed challenge-response defences suffice to ensure security of services and networks that rely on the correctness and availability of DNS.

We study DNS security and explore the threats that stem from the ubiquitious Internet connectivity, from the cyber arms-race and advances in adversarial capabilities, and from vulnerable systems. Our study 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 modern adversaries.

Given the lack of adoption of cryptograhic defences for DNS, understanding the security landscape of DNS without systematic defences is of critical importance.

We claim that DNSSEC is the most suitable defence for DNS against cache-poisoning attacks. Furthermore, as we show in this work, 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 goverments' agencies.

DOI 10.1109/SPW.2014.20

The main problem is that many of the attacks that do not break connectivity go undetected. For instance, recent hijacking of `google.rw` by the Syrian hackers, [10], 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 the whistleblower. DNSSEC would enable detection of such attacks.

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.

A. 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 required adversarial capabilities.

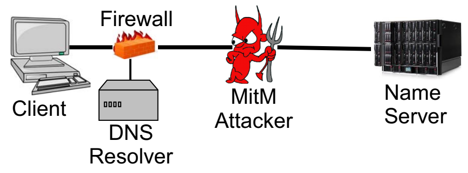

Fig. 1. Man-in-the-middle Attacker Model.

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 access 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).

Specifically, when accessing the Internet via untrusted networks, the clients typically use local recursive resolution services provided by the operators of those networks. Challenge-response defences are trivially not effective against a malicious resolution service. We demonstrate simple attacks exploiting vulnerabilities which stem from outsourcing DNS resolution services to untrusted network operators or from other malicious clients on those networks.

We also outline a critical privacy problem for clients accessing the Internet from untrusted public network, which enables attackers to track clients throughout their history of Internet access from different *networks*.

MitM on Backbone Links. Powerful adversaries, controlling routers located on backbone links have access to all traffic, and recent revelations, [9], show that there is a practice of injecting poisoned DNS responses into legitimate flows, to redirect clients to incorrect servers, e.g., for censorship.

MitM via Routing Hijacking. Inter-domain routing protocol, Border Gateway Protocol (BGP) [RFC4271], has known a history of route hijacking attacks, [11], 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, [12], and such hijacks may be frequently occuring.

B. 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) as well as in name servers' operating systems and DNS software. Our study shows that DNS infrastructure is still vulnerable to attacks.

C. Domain Name System Security Extensions (DNSSEC)

We argue that DNSSEC is the most suitable defence to thwart cache poisoning attacks. 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 defences for DNS, e.g., Eastlake cookies, [13], or DNSCurve, [14].

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.

Organisation

We review DNS and DNS cache poisoning in Section II. We then discuss threats from: (1) MitM adversaries and how attackers can obtain MitM capabilities (Section III) and (2) vulnerabilities in name servers and zones hosting infrastructure (Section IV). Finally, we discuss DNSSEC, its deployment challenges and application for forensic analysis (Section V).

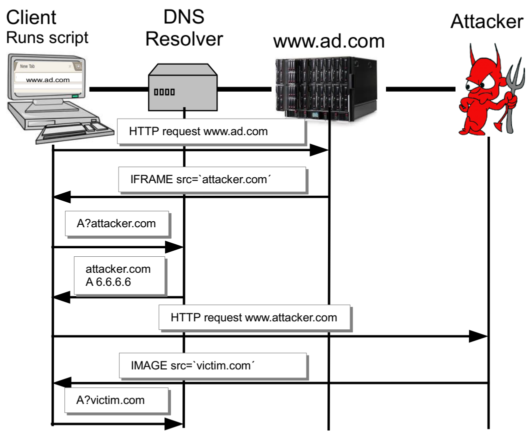

Fig. 3. Triggering DNS requests via a malicious script (puppet).

II. Domain Name System and Cache-Poisoning

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

A. *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 right-most 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.

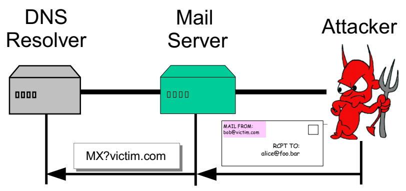

Fig. 4. Triggering DNS request by sending email to a victim network (simplified).

B. *DNS Cache-Poisoning*

In this section we provide background on DNS cache poisoning.

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 [15] to 30 million in 2013 [16]. 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) [17], 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, responses to which the (external) attacker wishes to poison.

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 [18].

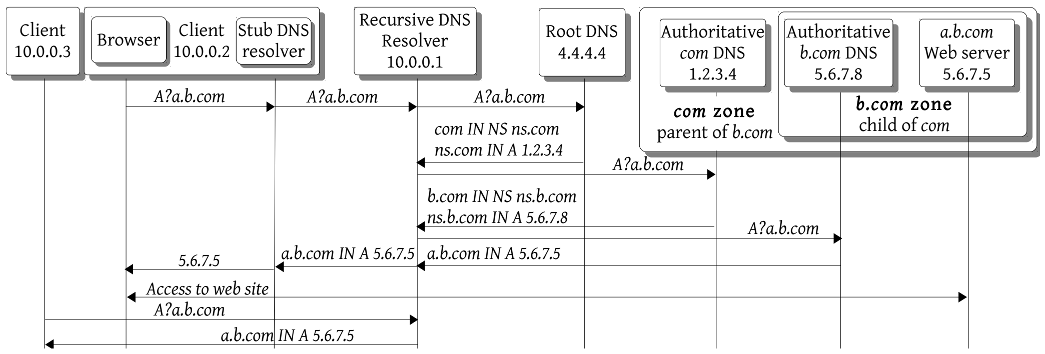

Fig. 2. Sample resolution process initiated by a stub resolver for an IP address of a web site. Recursive resolver performs the lookup and caches the resource records from the DNS response. Subsequent requests from clients for the same RRs are satisfied from the cache.

III. 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 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.

A. 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 Internet Service Providers (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.

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.e., packets sent to port 53, and to craft spoofed responses, before authentic responses arrive.

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.

B. MitM on Backbone Links

Recent revelations, [9], 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.

C. 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 Border Gateway Protocol (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, [11]. 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, [12], 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.

IV. 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 occuring. 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, bitdefender.com` were redirected.

A. 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.

B. 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.

V. DNS Security Extensions (DNSSEC)

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.

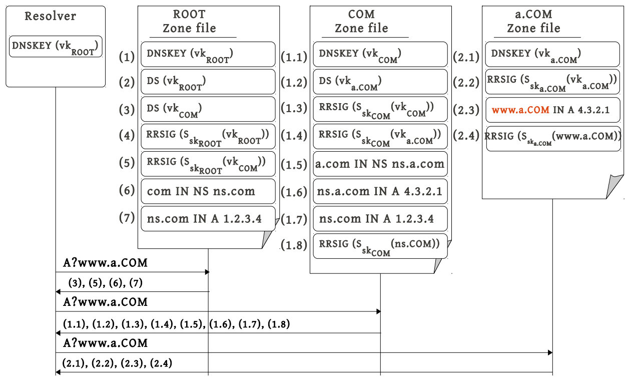
Resolver
DNSKEY (vk_{ROOT})
ROOT Zone file
(1) DNSKEY (vk_{ROOT})
(2) DS (vk_{ROOT})
(3) DS (vk_{COM})
(4) RRSIG ($S_{sk_{ROOT}}(vk_{ROOT})$)
(5) RRSIG ($S_{sk_{ROOT}}(vk_{COM})$)
(6) com IN NS ns.com
(7) ns.com IN A 1.2.3.4
COM Zone file
(1.1) DNSKEY (vk_{COM})
(1.2) DS ($vk_{a.COM}$)
(1.3) RRSIG ($S_{sk_{COM}}(vk_{COM})$)
(1.4) RRSIG ($S_{sk_{COM}}(vk_{a.COM})$)
(1.5) a.com IN NS ns.a.com
(1.6) ns.a.com IN A 4.3.2.1
(1.7) ns.com IN A 1.2.3.4
(1.8) RRSIG ($S_{sk_{COM}}$(ns.COM))
a.COM Zone file
(2.1) DNSKEY ($vk_{a.COM}$)
(2.2) RRSIG ($S_{sk_{a.COM}}(vk_{a.COM})$)
(2.3) www.a.COM IN A 4.3.2.1
(2.4) RRSIG ($S_{sk_{a.COM}}$(www.a.COM))
A?www.a.COM
(3), (5), (6), (7)
A?www.a.COM
(1.1), (1.2), (1.3), (1.4), (1.5), (1.6), (1.7), (1.8)
A?www.a.COM
(2.1), (2.2), (2.3), (2.4)

Fig. 5. 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.

A. Pitfalls of Inter-domain Dependencies

Inter-domain dependencies are common in DNS, and occur when a domain contains resource records in other domains. The dependencies stem from different motivations and goals, and are expressed, most notably, via NS, MX and CNAME records. When a DNSSEC signed zone depends on a non-signed zone, DNSSEC protection may fail; see [19]. This is especially relevant during incremental deployment, when DNSSEC is supported only by a fraction of the zones. The study carried out by [19] checked DNSSEC configuration of 14 industry unique US companies that adopted DNSSEC, according to a survey conducted by NIST, among 1070 domains [20]. Results were disappointing; [19] found that DNSSEC configuration of all but three of them, allowed DNS cache poisoning attacks of addresses of web servers (A), mail servers (MX) or name servers (NS); see Table I for summary of vulnerabilities.

B. Operational Challenges

There are a number of challenges related to deployment of DNSSEC, which we studied in our earlier work [21], [22]. 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 signing procedure. For instance, in January'12, Comcast (a large Internet Service Provider (ISP)) stopped serving responses for `nasa.gov`. This immediately incited spuculations 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'13, 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` top level domain (TLD). The impact was that 18 million clients of comcast Internet provider, 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.

C. 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 analsis, 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

Domain	Name Server	Mail Server	Web Server	Vulnerability
`datamtn.com`	✓	✓	✓	secure
`sprint.net`	✓	✓	✓	secure
`debian.org`	✓	✓	✓	secure
comcast.net	✓	✓	CNAME to unsigned domain	web site hijacking
comcast.com	✓	✓	CNAME to unsigned domain	web site hijacking
infoblox.com	✓	✓	CNAME to unsigned domain	web site hijacking
paypal.com	island-of-security	✓	CNAME to unsigned domain	web site and name server hijacking
nelnet.net	NS in unsigned domain	✓	CNAME to unsigned domain	web site and name server hijacking
ripe.net	NS in unsigned domain	✓	✓	name server hijacking
fedoraproject.org	✓	MX in unsigned domain	✓	mail server hijacking
iana.org	NS in unsigned domain	✓	✓	name server hijacking
icann.org	NS in unsigned domain	✓	✓	name server hijacking
ietf.org	NS in unsigned domain	✓	✓	name server hijacking
isc.org	NS in unsigned domain	✓	✓	name server hijacking

TABLE I. VULNERABLE DNSSEC CONFIGURATIONS AMONG US COMPANIES WHOSE DOMAINS ARE REPORTED, IN [20], AS SECURE DUE TO ADOPTION OF DNSSEC.

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.

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 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.

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 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, [13], and DNSCurve, [14].

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 earlier Sections III and IV, 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 Automous 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 Interent. The inconsistencies, if found, would be carefully checked to test for attacks.

VI. Conclusion

A secure DNS is critical for stability and functionality of the Internet. In this work we performed a study of DNS security and showed that the DNS infrastructure is vulnerable to attacks, and that many attacks are frequently launched by different adversaries. A signifcant problem pertaining to many attacks is that it is impossible to detect them (unless they break connectivity or ‘expected functionality).

We argue that DNSSEC could not only prevent most 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 attacks and breaches to third parties, insurance organisations, judges, Internet operators. 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 motivate adoption of systematic cryptographic defences, in particular, DNSSEC.

Acknowledgement

We are grateful for support for CAIDA's Internet Traces [23] that is provided by the National Science Foundation, the US Department of Homeland Security, and CAIDA Members. This work was supported by projects EC SPRIDE (BMBF) and CASED (LOEWE).

—

References

[1] A. Herzberg and H. Shulman, “Security of Patched DNS,” in *ESORICS*, ser. LNCS, S. Foresti, M. Yung, and F. Martinelli, Eds., vol. 7459. Springer, 2012, pp. 271–288. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33167-1

[2] ——, “Vulnerable Delegation of DNS Resolution,” in *European Symposium on Research in Computer Security*, ser. Lecture Notes in Computer Science. Springer, 2013.

[3] ——, “Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org,” in *CNS 2013. The Conference on Communications and Network Security. IEEE.* IEEE, 2013.

[4] ——, “Socket Overloading for Fun and Cache Poisoning,” in *ACM Annual Computer Security Applications Conference (ACM ACSAC)*, December 2013.

[5] H. Shulman and M. Waidner, “Fragmentation Considered Leaking: Port Inference for DNS Poisoning,” in *Applied Cryptography and Network Security*. Springer, 2014, pp. –.

[6] P. Neira, “Patchset of Netfilter Updates,” http://patchwork.ozlabs.org/patch/307041/, 2013.

[7] H. F. Sowa, “ipv4: introduce new ip mtu discover mode ip pmtudisc interface,” Linux Kernel Mailing List, 2013, http://thread.gmane.org/gmane.linux.network/288482.

[8] National Security Agency, “PRISM (surveillance program),” 2007.

[9] ——, “FOXACID and QUANTUM Programs,” 2013.

[10] F. Denis, “The GOOGLE.RW Hijack,” http://labs.umbrella.com/2013/10/25/google-rw-hijack-nobody-else-noticed/, 2013.

[11] H. Ballani, P. Francis, and X. Zhang, “A Study of Prefix Hijacking and Interception in the Internet,” in *ACM SIGCOMM Computer Communication Review*, vol. 37. ACM, 2007, pp. 265–276.

[12] K. Bode, “Somebody is Hijacking Massive Amounts of Data Via Iceland,” http://www.dslreports.com/shownews, 2013.

[13] D. Eastlake, “Domain Name System (DNS) Cookies,” https://tools.ietf.org/html/draft-eastlake-dnsext-cookies-04, 2014.

[14] D. J. Bernstein, “DNSCurve: Usable security for DNS,” Posted at: http://dnscurve.org/, 2010.

[15] D. Leonard and D. Loguinov, “Demystifying service discovery: implementing an internet-wide scanner,” in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 109–122.

[16] “Open DNS resolver project,” last retrieved May 2013. [Online]. Available: http://openresolverproject.org/

[17] S. Antonatos, P. Akritidis, V. T. Lam, and K. G. Anagnostakis, “Puppetnets: Misusing Web Browsers as a Distributed Attack Infrastructure,” *ACM Transactions on Information and System Security*, vol. 12, no. 2, pp. 12:1–12:15, Dec. 2008.

[18] A. Herzberg, “DNS-based email sender authentication mechanisms: A critical review,” *Computers & Security*, vol. 28, no. 8, pp. 731–742, 2009.

[19] A. Herzberg and H. Shulman, “Retrofitting Security into Network Protocols: The Case of DNSSEC,” *Internet Computing, IEEE*, vol. 18, no. 1, pp. 66–71, 2014.

[20] National Institute of Standards and A. N. T. D. Technology, “Estimating Industry IPv6 and DNSSEC External Service Deployment Status,” http://fedv6-deployment.antd.nist.gov/cgi-bin/generate-com.

[21] A. Herzberg and H. Shulman, “Dnssec: Interoperability challenges and transition mechanisms,” in *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*. IEEE, 2013, pp. 398–405.

[22] ——, “Dnssec: Security and availability challenges,” in *Communications and Network Security (CNS), 2013 IEEE Conference on*. IEEE, 2013, pp. 365–366.

[23] CAIDA, “Anonymized Internet Traces 2012 Dataset,” http://www.caida.org/data/passive/passive_2012_dataset.xml, 2012.