Towards Adoption of DNSSEC: Availability and Security Challenges

Amir Herzberg and Haya Shulman
Computer Science Department
Bar Ilan University
Email: {amir.herzberg,haya.shulman}@gmail.com

***Abstract*—DNSSEC deployment is long overdue; however, it seems to be finally taking off. Recent cache poisoning attacks motivate protecting DNS, with strong cryptography, rather than with challenge-response 'defenses'.**

Our goal is to motivate and help correct DNSSEC deployment. We discuss the state of DNSSEC deployment, obstacles to adoption and potential ways to increase adoption. We then present a comprehensive overview of challenges and potential pitfalls of DNSSEC, well known and less known, including:

- ***Vulnerable configurations:* we present several DNSSEC configurations, which are natural and, based on the limited deployment so far, expected to be popular, yet are vulnerable to attack. This includes NSEC3 opt-out records and inter-domain referrals (in `NS`, `MX` and `CNAME` records).**
- ***Incremental Deployment:* we discuss potential for *increased vulnerability* due to popular practices of incremental deployment, and recommend secure practice.**
- ***Super-sized Response Challenges:* DNSSEC responses include cryptographic keys and hence are relatively long; we explain how this extra-long responses cause interoperability challenges, and can be abused for DoS and even DNS poisoning. We discuss potential solutions.**

Keywords: DNSSEC, DNS security, DNS cache poisoning.

I. Introduction

The correctness and availability of information in the Domain Name System (DNS) are crucial for the correct and secure operation of the Internet. However, there is a long history of attacks on the DNS, most notably *DNS cache poisoning*, [1], [2], [3], [4], where an attacker provides fake mappings in responses to DNS requests. Since DNS uses caching for performance, fake mappings can be cached and used for long time and to attack many systems.

DNS cache poisoning is a significant threat to Internet security: it may allow weak *off-path* attackers to redirect communication to incorrect, adversarial, servers, thereby enabling the off-path attacker to intercept and modify content. DNS cache poisoning allows to circumvent many defense mechanisms such as Same Origin Policy (SOP), domain blacklists and domain-policies (e.g., SPF), exposing to a range of attacks, such as phishing, credentials-theft, and more.

The importance of correct functionality of DNS, in tandem with a long history of attacks, yielded significant efforts to improve DNS security. Defenses against DNS poisoning, can be categorised into two classes: *challenge-response* defenses, where resolvers add random challenges to requests and validate them in responses, [5], and *cryptographic* defenses, most notably DNSSEC, [6], [7], [8].

Cryptographic defenses ensure security even against a MitM attacker, who is able to eavesdrop and modify traffic; however, they require adoption by both end-points to the DNS transaction and significant deployment and maintenance efforts. In contract, challenge-response mechanisms are relatively easy to deploy and maintain, e.g., they require support *only* by the resolver, but are not secure against MitM attacker. Currently, most DNS resolvers rely on challenge-responses defenses for their security.

However, recently practical attacks were shown, [1], [2], [4], allowing even an off-path attacker to circumvent the challenge-responses mechanisms, exposing DNS to efficient cache poisoning attacks. We hope and believe, that the publication of these attacks, will catalyze the adoption of DNSSEC.

In this paper, we try to further encourage deployment of DNSSEC, by discussing the design, motivations, status and challenges. Furthermore, we identify potential pitfalls which may result in deployment challenges, or, worse, in vulnerabilities, and discuss countermeasures.

A. Challenge-Response Defenses

Challenge-response mechanisms attempt to provide security against off-path adversaries, by using some 'unpredictable challenge' values. The DNS resolver should select the challenge values (at random) and send them within the requests. Then, the resolver should validate that the corresponding responses, from the intended name server, echo those values. Two challenge-response mechanisms are adopted by most resolvers: the DNS transaction identifier (TXID), and the source port field (added in 'patches' following the Kaminsky attack [1]). Additional mechanisms were proposed and adopted by some resolvers, e.g., random challenges embedded within the query (random prefix and case randomization); see, e.g., [5], [9].

All challenge-response defenses are ineffective against a man-in-the-middle (MitM) attacker: a MitM attacker can simply copy the values from the requests to (spoofed) responses. However, (until recently) the folklore belief was that challenge-response defenses were effective against *off-path* attackers: an off-path attacker does not receive the request, and hence has to guess the challenge values in its attempt to construct a (spoofed) response. By using sufficient entropy in the challenges, guessing the challenge fields become impractical. In particular, both the TXID and the source port

are 16-bit fields; hence, the TXID alone provides insufficient entropy (exploited in [1]), but as observed there, if both are truly random, the challenge has 32 bits of entropy, which may often suffice to render Kaminsky's attack impractical.

However, in [2], [4], we presented different techniques that allow off-path attackers to circumvent different challenge-response defenses, and predict the challenge values, rendering DNS resolvers exposed to efficient cache poisoning attacks. While it is possible to prevent these recent attacks, surely they provide additional motivation to adopt DNSSEC.

B. DNSSEC: Cryptographic Defense

DNSSEC [8], [7], [6] provides security against MitM adversaries by relying on cryptographic authentication (signature) of records in responses. Its main disadvantage, compared to most challenge-response defenses, is that it requires adoption both by the DNS zones and resolvers. Indeed, although DNSSEC was proposed in 1997 and throughly evaluated as well as validated analytically [10], it is still not widely supported and deployed.

For example, we found that only about 2% of the most popular[1] domains are signed. Furthermore, the measurements in [12] found that about 68% of the resolvers request DNSSEC records, but only about 1% of them actually validate the responses. Apparently, the vast majority of resolvers that signal support of DNSSEC, are in fact *non-validating*, i.e., do *not* reject responses which are not properly signed, e.g., using the *permissive validation mode* in Unbound 1.4.19. Ironically, as we recently showed [4] (and summarize briefly in Section V-B), this large percentage of resolvers that request DNSSEC records but do not validate them, may often become vulnerable to off-path attacks.

This limited deployment of DNSSEC is disappointing, considering its importance and extensive support by the Internet and security communities. Much of the existing efforts, e.g., the Internet Society Deploy360 program [13], focus on educating consumers, domain-owners, infrastructure and software providers, to increase support for DNSSEC. In this paper, we focus on the complementing technical aspects, including challenges, vulnerabilities, and recommended practices.

It seems that significant additional effort is required to make DNSSEC fully functional, in most domain names and resolvers; and that more care must be taken to ensure correct deployment.

Contributions

The main contribution of this work is in encouraging adoption of DNSSEC, while at the same time, raising awareness to subtleties and potential vulnerabilities, to make sure that deployements are indeed secure. We investigate the following issues related to DNSSEC: *deployment and interoperability issues*, *vulnerable DNSSEC configurations*, and *attacks on oversized DNS responses*.

[1]We computed the statistics for the top 300,000 most popular domains as listed by Alexa [11].

a) Deployment and Interoperability: We review the current deployment status of DNSSEC and obstacles to deployment, most notably the interoperability problems of DNSSEC-enabled responses with intermediate Internet devices. We discuss the common practice of incremental deployment, and show that it may cause *increased* vulnerability to poisoning.

b) Vulnerable DNSSEC Configurations: we present several DNSSEC configurations, which are natural and, based on the limited deployment so far, expected to be popular, yet are vulnerable to attack. This includes NSEC3 opt-out records and inter-domain referrals (in `NS`, `MX` and `CNAME` records).

c) Attacks on Oversized DNS Responses: DNSSEC responses include cryptographic keys and hence are relatively long; we explain how this extra-long responses cause interoperability challenges, and can be abused for DoS and even DNS poisoning. We discuss potential solutions.

Organisation

We provide a background on DNS and DNSSEC in Section II, and discuss deployment status and interoperability problems in Section III. We review vulnerable DNSSEC configurations in Section IV, and exploits of large, DNSSEC-enabled, DNS responses in Section V. We then conclude this review of DNSSEC in Section VI.

II. Background

In this section we provide an overview of DNS, and its extension mechanisms: EDNS and DNSSEC.

A. Domain Name System

The domain name system (DNS), [14], [15], 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 organized 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 perform respond 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 TTL field of a response. Subsequent requests for the same RRs are provided from the

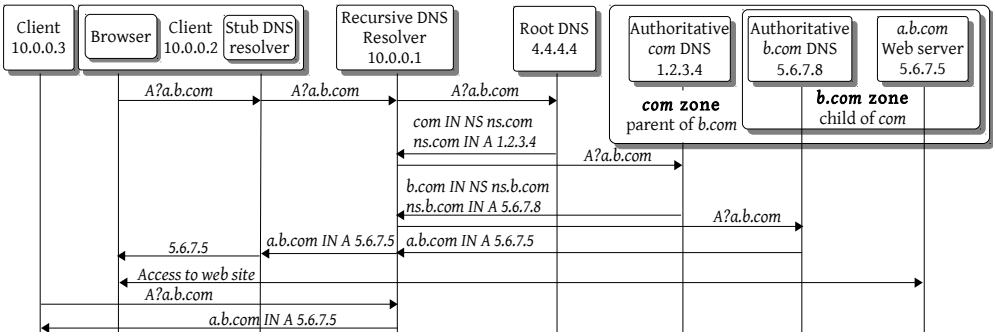

Fig. 1. Sample resolution process initiated by a stub resolver for an IP address of a web site. Recursive resolver performs the lookup and caches the resource records from the DNS response. Subsequent requests from clients for the same RRs are satisfied from the cache.

cache.A sample lookup process, initiated with a DNS request from a stub resolver, is depicted in Figure 1.

B. Extensions to DNS (EDNS)

The original DNS specifications fix the maximal size of a DNS packet, when sent over UDP, to 512 bytes (RFC 1035). Longer responses were unusual and *truncated*, by returning a partial response and signaling truncation (TC bit set). Upon receiving truncated response, resolvers should resend the request over TCP.

However, using TCP for DNS requests imposes significant overhead, requires state in the name server, and adds latency to responses. Support of longer responses, e.g., for DNSSEC, without such overhead, is possible using the *Extension Mechanisms for DNS (EDNS)* [16].

EDNS is an optional, but widely deployed, mechanism, that allows clients to advertise certain capabilities to DNS servers. One of those capabilities is a larger reassembly buffer. For example, a DNS resolver can advertise that it can reassemble a 2000 bytes DNS response, which is significantly larger than the legacy limit of 512 bytes. This provides support for larger DNS responses, as required to transmit (over UDP) long, DNSSEC-enabled responses, containing keys and signatures.

C. DNS Security (DNSSEC)

When no protection is employed, DNS requests and responses can be inspected and altered by a MitM attacker. For example, a malicious wireless client can tap the communication of other clients and can respond to their DNS requests with maliciously crafted DNS responses, containing a spoofed IP address, e.g., redirecting the clients to a phishing site.

Domain Name System Security Extensions (DNSSEC) standard [6], [7], [8] 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.

As an example, consider the (simplified presentation of) chain of trust constructed in Figure 2. The resolver has a public verification key vk_{ROOT} of the root, and it needs to

obtain the A RR (an IP address) for host *www.a.com*, which is in a *a.com* zone. The resolver queries the root, and obtains the DS and NS RRs (and corresponding IP address in A RR) for *com* zone; this data tells the resolver that it should query the name server authoritative for *com* zone. DNSKEY of the root (vk_{ROOT}) is used to verify the signature $S_{sk_{\text{ROOT}}}(vk_{\text{COM}})$ on the DS RR of the child, i.e., *com* zone. Subsequently, *com* zone sends the resolver the DS RR of the child zone, i.e., *a.com*. The DNSKEY vk_{COM} of the parent zone *com* is used to verify the signature $S_{sk_{\text{COM}}}(vk_{\text{A.COM}})$ on the DS RR of the child *a.com*. The chain of trust terminates with a DNSKEY RR whose corresponding private key $sk_{\text{A.COM}}$ signs the requested DNS data.

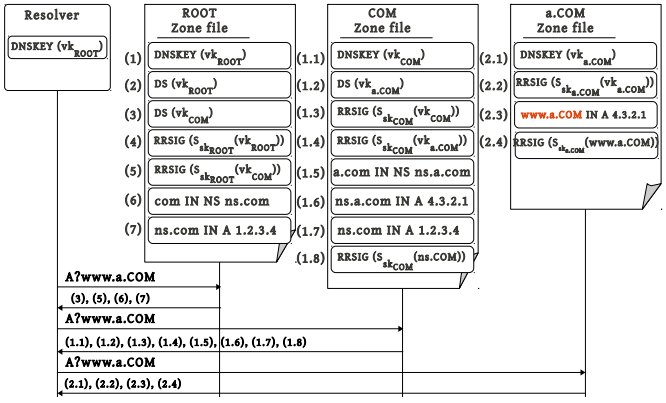

Fig. 2. A (simplified) sample process of constructing a chain of trust from the root zone to the public verification key $vk_{\text{A.COM}}$ of the target zone *a.com*; the name servers are depicted with their corresponding zone files, that (for simplicity) contain only the RRs relevant to DNSSEC. For 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.

Authenticated Denial of Existence: When a DNS request specifies a non-existing record, the name server responds with a *non-existing domain (NX-Domain)* response. To authenticate such responses, DNSSEC does not send a signature over the non-existing domain name, since this required real-time signing, contradicting DNSSEC's strategy of only signing off-line for better security and performance (see above).

Instead of signing each non-existing response, DNSSEC includes alternative authentication mechanisms to allow a name server to prove non-existence of a resource: the NSEC and NSEC3 record types.

NSEC, [7], allows to (cryptographically) prove that a resource record set does not exist, by spanning a gap between two domain names in a zone. NSEC specifies what type of records exist at a name where it resides and points to the next domain name (in canonical order) in the zone. One obvious drawback of NSEC is that it allows discovery of all subdomains, since each NSEC record points at the next domain name.

To fix the subdomain exposure risk, an alternative was proposed: NSEC3, [17] is a chain of hashed names that should prevent enumerability. Subsequently, in 2009 Bernstein showed that NSEC3 can also allow discovery of subdomain names.

NSEC3 supports supports *opt-out* option, where only secure delegations have NSEC3 record (i.e., delegations to subdomains that are signed). Opt-out NSEC3 allows better performance for large domains with many unsigned subdomains, and hence is widely used and often recommended [18]. Opt-out allows shorter NSEC3 chains with fewer signatures and smaller signed zone files.

III. Deployment and Interoperability

In this section we review the status of DNSSEC deployment and validation at resolvers and name servers, discuss deployment challenges and solutions.

A. Interoperability Problems

Intermediate devices, on the path between the resolver and the name server, may prevent DNSSEC service, by blocking EDNS queries/responses, blocking long and/or fragmented responses, or removing DNSSEC records from DNS responses; see [19], [20], [21]. These interoperability problems are due to differences between DNSSEC enabled DNS responses and 'legacy' DNS responses, mainly, support for EDNS, special RR types, and long responses. For instance, consider Figure 3, which shows the sizes of the responses of sub-domains of *gov*, one of the top level domains with highest adoption of DNSSEC (by sub-domains). The responses from more than a half of the domains in *gov* are of size larger than 1500 bytes, when queried for an ANY RR, and a significant fraction also have A RR responses of length over 1500 bytes. The vast majority of these long responses are due to the use of DNSSEC by *gov* and (many of) its sub-domains.

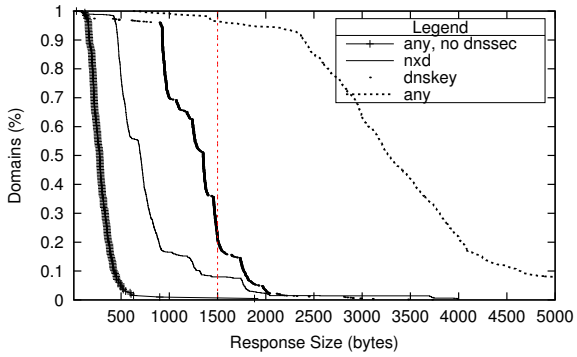

Fig. 3. Length of responses for ANY and A RR queries of GOV domains. Domains taken from [22]

Why are DNSSEC responses so much longer, and what are the other differences? When sending a DNS request the client adds an EDNS OPT pseudo RR to the query, and sets the DO bit, indicating support of DNSSEC; the client also indicates the maximal response size in the advertised EDNS buffer size. The response from DNSSEC-compliant name server, receiving such request for DNSSEC-enabled domain, responds with a lo

Security enabled name server, receiving such a request, add to the response DNSSEC RRs for signatures, keys, etc..

A DNS response containing DNSSEC RRs often exceeds the 512B maximal DNS size specified in RFC1035 [23]. Some firewalls drop such 'non-conforming' DNS packets. Furthermore, such responses may exceed the maximal transmission unit (MTU) on the path, and may thus get dropped or fragmented. Fragments are blocked by some firewalls, mainly for security reasons. In addition, some DNS proxies may not support EDNS and/or not support (and drop) the DNSSEC record types [21]. As a result of all this, DNS responses may not reach the client.

To avoid interoperability problems due to long responses, domains may use shorter public keys and signatures; this is also recommended, e.g., by NIST [24]. Indeed, the use of 1024 bit RSA keys seem common, and one motivation is probably to reduce response size. Unfortunately, 1024 bit RSA is already considered not secure enough, e.g., by NIST [25]. Notice that sufficient security with short keys is possible with appropriate cryptosystems, e.g., elliptic curve DSA, standardized for DNSSEC [26]; however, this algorithm is implemented only by a negligible number of domains (0.002%) [27].

In an experimental study of interoperability issues by Nominet [19], only 4 out of 24 tested units (broadband router/firewalls) could support DNS responses larger than 512 byte. Establishing a TCP connection would solve the problem of the size of DNS responses; however, the client will never attempt to connect via TCP if it does not receive a (truncated) response from a DNS server, e.g., say if it were blocked due to fragmentation. Furthermore, [19] found that support for TCP in router/firewall DNS proxies (which they tested) is almost non-existent, i.e., just one unit out of 24 could proxy TCP.

Even when the length of the responses is acceptable, there may be problems due to the DNSSEC resource records, e.g., the EDNS OPT pseudo 'extension RR' [28] used to carry the signaling information, or the signatures carried in RRSIG RRs. According to [19], 6 out of 22 devices did not handle correctly DNS queries containing the EDNS RR. The result was that either plain DNS RRs are returned (without DNSSEC records) without indication of error, or the query times-out.

Due to concerns about interoperability issues, many resolvers that support DNSSEC, allow unvalidated responses, thereby allowing *downgrade attack* [8], where an attacker sends fake responses that appear similar to responses passing through non-interoperable devices. For instance, consider an attacker that poisons the NS and glue A RRs, and provides a forged delegation NS RR to redirect the clients to a host of attacker's choosing; recall that the NS and A RRs are defined to be the child zone's authoritative data and they are kept unsigned in the parent zone. Since the attacker does not have the secret signing key of the child, it cannot forge signatures on the DNS data, but it does not need to, when the resolver will revert to plain DNS without DNSSEC protection.

B. DNSSEC Deployment

Most DNS resolvers do not support DNSSEC validation, and many domains are not signed with DNSSEC. The fact that there is currently little support of DNSSEC further reduces a motivation for early adopters, since protection of DNSSEC only 'kicks in' when all the entities, involved in a resolution of some domain, support DNSSEC.

1) DNSSEC Validation at Resolvers: A significant fraction of the resolvers currently signal DNSSEC support; however, *less than 1%* actually *enforce DNSSEC validation* [27], [12], [29]. Obviously, for such resolvers, DNSSEC does not provide added security. This approach apparently assumes that permissive use of DNSSEC can provide evidence on whether the network can deploy DNSSEC fully without problems or not, while not *harming* their security; however, in fact, such resolvers are open to poisoning, and may even facilitate the attack of [4].

2) DNSSEC Deployment at Zones: To make DNSSEC validation effective in resolvers the domains have to adopt DNSSEC. However, most do not. DNSSEC adoption by top level domains is not bad; the root and the largest domains such as *com, net* and *org* support DNSSEC, and we found that currently 30% of the TLDs use DNSSEC, apparently mostly the larger.

DNSSEC adoption is significantly worse for (important) second-level domains. We found that currently only 2% of the top 300,000 domains (according to Alexa [30]) support DNSSEC. Even worse results were found by NIST [31], who checked DNSSEC adoption in 1070 large industrial US domains; out of them, they found only 14 (1.4%) have adopted DNSSEC. Worse: as we show in Table I, only three out of these 14 domains are really secure; see details in Section IV-A.

Islands of security play an important role in the lack of deployment of DNSSEC. An island of security is a signed zone that does not have a DS RR at its parent, e.g., since the parent is not signed. Thus the resolver cannot construct an authentication chain leading down from the trust anchor to the target DNSSEC-enabled zone; as a result, resolvers cannot authenticate the DNSKEY RR of the child zone.

Currently, there are many islands of security; according to [32], islands of security constitute 76.6% of the total number of DNSSEC enabled zones on the Internet.

To facilitate distribution of trust anchors of islands of security, *Trusted Anchor Repositories (TARs)* were proposed, e.g., [33]. The TAR maintains the public verification keys of islands of security. However, the TARs are not widely supported since the zones are required to *add* their public keys to TARs and resolvers should be *configured* to query the TAR to obtain the key when needed.

IV. Vulnerable DNSSEC Configurations

In this section we review two issues which introduce vulnerabilities in the protection offered by DNSSEC: one is related to *inter-domain dependency* of DNS and the other is related to *non-deniability of existence* of DNSSEC.

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. As we next explain, such inter-domain dependencies may limit the effectiveness of the protection offered by DNSSEC against cache poisoning, especially during incremental deployment, when the DNSSEC is only partially supported by domains on the Internet. We conducted a study (Table I) of DNSSEC configuration by Industry unique US companies that adopted DNSSEC, according to survey of 1070 domains by NIST [31]; we found that the DNSSEC configuration, that most of them support, allows DNS cache poisoning attacks of addresses of web servers (A), mail servers (MX) or name servers (NS). We next discuss the vulnerabilities pertaining to dependencies via different records, and which exploits they allow.

1) Dependency via NS records: To find the name server of a subdomain, e.g., $foo.bar$, the resolver makes a query to $ns.bar$, the name server of the parent domain (bar), which responds with a *referral*: the name server $ns1.foo.bar$ of the name server(s) of $foo.bar$. It is possible, that a name server of $foo.bar$ will be in a different domain, e.g., $fie.baz$. Even if both $foo.bar$ and its parent domain bar use DNSSEC, the referral sent by $ns.bar$ is not signed, since in DNSSEC, a name server only provides signatures for its own domain.

Bernstein observed that this implies, that a MitM attacker can spoof such referral responses [34]. However, [10] observed that this cannot allow DNS poisoning, since that validating resolvers should not follow delegation responses without a signed DS record (when a domain is signed) from the parent zone. The intuition is that if the name server is in the target zone then the resolver expects to receive signed NS and A records from the name server of the child zone. If it does not, it should not trust the NS and A received in the referral from the parent zone.

However, this argument is not relevant when the name server (of a DNSSEC protected domain), is in a domain NOT protected by DNSSEC. This is the case, for example, for six (!) of the 14 DNSSEC-adopting domains in Table I. In such cases, a MitM attacker - and in some cases, often *due* to the use of (non-validated) DNSSEC - may be able to send wrong mappings for the name server (and then, trivially, for specific domains too).

Furthermore, [35] noted that this mechanism also has security implications. For example, when domain $foo.bar$ specifies a name server in another domain, say $ns.foe.baz$, then the mappings of $foo.bar$ can be controlled (and altered!) not only by the administrators of bar and $foo.bar$, but also by the administrators of baz, $foe.baz$ and $ns.foe.baz$; we say that domain $foo.bar$ trusts domain $foe.baz$. Furthermore, supposed domain $foe.baz$ has name servers in another domain, say $ns1.fee$, i.e., $foe.baz$ trusts domain fee. Then, domain $foo.bar$ also depend on (i.e., trusts) domain fee; [35] call this the *DNS transitive trust* property, and showed that it can be extensive - a typical name depends on 46 servers on average - and has security implications: compromise of a single name server on the resolution chain can lead to domain hijacks.

Not signing the NS or A records in referral responses does not allow spoofing responses from the authentic name servers (that serve signed records). However, spoofing the name server records results in a denial-of-service via DNS cache poisoning: the resolver caches the spoofed records. As a result, even when a zone appears to be properly signed, and a resolver can establish a chain of trust to that zone, it may still be vulnerable.

For example, consider the name servers used by paypal.com, which are not in the same domain as paypal; see Figure 4. The parent zones serve NSEC3 for those domains, indicating that they are not signed[2]. Therefore, although the resolver can establish a chain-of-trust to paypal.com it is still susceptible to DNS cache poisoning attacks due to the transitive trust property of DNS.

```
paypal.com.             300     IN      NS      ns2.isc-sns.com.
paypal.com.             300     IN      NS      ns1.isc-sns.net.
paypal.com.             300     IN      NS      ns3.isc-sns.info.
```

Fig. 4. Impact of transitive trust on the effectiveness of DNSSEC: paypal.com supports DNSSEC, however, its name servers are in domains outside of paypal.com. The parent zones of the name servers serve NSEC3 records, which indicates that they do not support DNSSEC; the illustration is a screen capture of *dig*.

```
www.paypal.com: type CNAME, class IN, cname www.paypal.com.akadns.net
www.paypal.com: type RRSIG, class IN
www.paypal.com.akadns.net: type CNAME, class IN, cname wlb.paypal.com.akadns.net
wlb.paypal.com.akadns.net: type CNAME, class IN, cname www.paypal.com.edgekey.net
www.paypal.com.edgekey.net: type CNAME, class IN, cname e6166.b.akamaiedge.net
e6166.b.akamaiedge.net: type A, class IN, addr 23.44.242.234
```

Fig. 5. Impact of transitive trust on the effectiveness of DNSSEC: paypal.com supports DNSSEC, however, the web site www.paypal.com is mapped (via a CNAME chain) to a web site hosted in an unsigned domain e6166.b.akamaiedge.net; the illustration is based on a collection of response packets from wireshark.

2) Dependency via CNAME to Unsigned Domain: Another type of dependency is mapping a resource record, via a CNAME (alias), to name in a different domain; CNAME is typically used to map services, e.g., ftp or web servers, to different names. This is useful for, e.g., web hosting, content delivery networks (CDNs).

Consider again paypal.com. This zone is signed and serves properly signed keys enabling the resolvers to validate the DNS records. However, in order to reach the machine hosting www.paypal.com the client has to traverse a lengthy CNAME chain; see Figure 5 that presents a collection of records taken from responses captured by wireshark. The first CNAME delegation is signed, while the subsequent four others, are *not*.

[2]Note that these zones (in Figure 4) use a DLV to allow resolvers to retrieve their signatures, however, this requires explicit configuration and is not supported by most resolvers.

As a result, although the domain www.paypal.com is signed, the actual IP address 23.44.242.234 of the web server hosting the web page of paypal is not signed. This enables MitM attackers to replace the mapping to a different IP address, which would be accepted and cache by validating resolvers.

As a result, clients accessing the web site www.paypal.com, would be redirected to attacker's controlled host, without any failure from DNSSEC validation; see Table I for a list of domains, among the domains that deployed DNSSEC [31], that are vulnerable to this type of attack.

3) Dependency via MX to Unsigned Domain: A dependency on an unsigned domain can be via an MX record, whereby a mail server is mapped to a different name. This allows cache poisoning the mail server. Notice that the implications of mail server hijacking are severe, and can be used for a range of attacks, including for credentials theft by exploiting password recovery via email as proposed by Kaminsky [1].

B. Pitfalls of Non-Deniability of Existence

Bernstein, [36], pointed out that NSEC is exposed to zone enumeration via zone walking the NSEC chain. To fix the zone walking an alternative was proposed: NSEC3, [17], whihc is a chain of hashed names that should prevent enumerability. Subsequently, Bernstein, [34], showed that NSEC3 also leaks private data allowing zone walking.

Recently, Bau and Mitchell [10] presented attacks allowing subdomain injection for domains supporting NSEC3 with opt-out. Opt-out DNSSEC requires only secure delegations to have NSEC3 record (i.e., delegations to subdomains that are signed).

V. Attacks on Oversized DNSSEC Responses

As discussed in Section III and illustrated in Figure 3, the use of DNSSEC results in long DNS responses, mainly due to the inclusion of (multiple) long records containing cryptographic keys and/or signatures. In the following subsections, we discuss two potential abuses of such 'oversized', long responses: their abuse for Denial-of-Service (DoS) amplification attacks, and their abuse to manipulate responses, in particular, for poisoning.

The problems of oversized DNSSEC responses are mainly due to the fact that DNS responses are normally sent over UDP. Some servers, most notably those serving large top-level domains `com` and `net`, send long responses over TCP rather than over UDP. Since TCP connection begins with a three-way handshake, attackers cannot complete the connection using spoofed source IP address (recent TCP injection attacks, such as [37], are inapplicable). Notice, however, that the use of TCP increases overhead on both network and server, for all responses - which can be significant. Furthermore, if not properly prevented, the use of TCP may even allow SYN-clogging attack against the server, see, e.g., [38].

In subsection V-C we argue that part of the problem is due to the design of DNSSEC, which results in potential unnecessary transmission of unrequired keys and signatures, esp. if domains decide to support multiple keys and algorithms (as is highly desirable), and sketch potential (long-term) solutions.

A. Reflection Amplification Attacks

When a domain supports DNSSEC, it returns, in addition to traditional DNS records, also DNSSEC records, for cryptographic keys and signatures. As a result, the size of DNSSEC-enabled DNS responses significantly exceeds the size of traditional DNS responses; see Figure 3. Such responses are often abused by attackers to launch *amplification DoS attacks* to clog victim networks and hosts, see, e.g., [39].

In a DNS amplification DoS attack, the attacker sends to one or more DNS servers many requests, with spoofed (fake) source IP address. Name servers respond to such requests by sending (much longer) responses to the IP address that originated the DNS request. The *amplification factor* is the ratio between the number of response bytes sent by the *amplifying* (benign) DNS server, to the number of request bytes sent by hosts controlled by the attacker, in the corresponding requests. With DNSSEC, the ratio can be as high as a hundred. Indeed, while DNSSEC deployment is still very limited, it has already been abused in the largest DoS attacks in the recent years, with reported bandwidths of 100 Gbps (2010), 60 Gbps (2011, 2012) [40] and 300 Gbps [41].

Sometimes, the victim of the DNS amplification attack is the amplifying DNS server itself, which wastes computation and network resources on sending the (amplified) response packets; however, more often, the victim is a network or host, which receives the responses. The DNS servers use various mechanisms, to prevent their abuse as vectors in amplification attacks, in particular, to avoid wasting their own resources due to such attacks. In fact, the concern about amplification attacks, may be one of the considerations against adoption of DNSSEC by name servers. Other defenses for name servers against their abuse as amplifiers include:

- Imposing a maximal *quota* of queries from the same source IP address. This defense may help against attacks against a remote host or network, using the same (or related) IP addresses, but not against attacks on the amplifying server itself, where the source IP address is not important. Furthermore, this defense fails if there are many spoofed IP addresses, connected via the same victim network, as in the Coremelt attack [42]. Finally, DNS request quotas are prone to false-positives, i.e., identification of a benign sending resolver as an attacker.
- Challenge-response mechanisms, mostly based on redirection of the request to another domain name, where the new domain name includes a random challenge ('nonce') used to ensure that the request did not contain a spoofed IP address. However, existing designs, e.g., [43], [44], do not support DNSSEC validation.
- As mentioned above, some servers, most notably those serving the large top-level domains `com` and `net`, defend against this threat by sending long responses over TCP rather than over UDP.

The above defenses are relevant only when many requests are sent to the same DNS server, e.g., against attacks of that server. They are irrelevant, when the goal of the attacker is

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	name server hijacking
nelnet.net	NS in unsigned domain	✓	CNAME to unsigned domain	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 [31], AS SECURE DUE TO ADOPTION OF DNSSEC.

to clog some victim network with the response packets, and the attacker distributes the requests between many different servers. As awareness to DNSSEC increases and its adoption speeds up, such amplification reflection attacks may become feasible, by sending requests to name servers of many different domains.

However, even currently, this attack is applicable, by sending requests to *open DNS resolvers*; latest measurements found over 25 millions to be exploitable for amplification attacks [45], [46]. Indeed, DNS amplification using open resolvers facilitated most of the large DoS attacks in recent years [40], [41]. Considering that most of these open resolvers are not well maintained, it seems futile to hope that they will implement anti-amplification defenses; and considering their vast numbers, blacklisting and filtering traffic from them by routers, may not be feasible. However, deployment of DNSSEC is not really necessary for this attack, since attackers could use other long DNS responses, as well as establish their own DNS domains with long responses.

B. Off-path attacks on fragmented responses

DNSSEC responses are often long enough, that they can exceed the MTU of networks in the path from the name server to the resolver, and hence, get *fragmented* by the name server or by a router along the path. The fragments are forwarded, as regular IP packets, to the destination (resolver), where fragments are kept in the *defragmentation cache* until all fragments of a packet are received, and the packet is reconstructed.

Ironically, as we recently showed [4], an attacker can send spoofed fragments, tricking the defragmentation mechanism into reassembling them together with the fragments sent by the legitimate source (name server). Conceptually, this technique is simple: the spoofed fragments just need to have the same source and destination IP addresses and the same 16-bit IP-ID (IP packet identifier), as the 'real' fragments. The challenge is to find the IP-ID, and to send spoofed fragments. In [4], we show that this requires only off-path capabilities, i.e., attacker does not need eavesdropping (or MitM) capabilities.

By sending random spoofed fragments, the attacker can easily cause corruption of the original packets; this can be used directly as a DoS attack, disrupting resolver to name server communication, or to cause the resolver to select other name servers, possibly more convenient to the attacker, as in [2].

Furthermore, due to the currently-common case of permissive validation, the off-path attacker will even be able to perform cache poisoning. In the case of permissive DNSSEC validation, DNS poisoning only requires a valid DNS response, with the correct port, DNS-identifier and query; luckily for the attacker, all these are contained in the *first fragment*. The attacker can send a spoofed second fragment, combining it with the legitimate first fragment, to successfully perform DNS poisoning; see [4] for details.

C. Unnecessary records in DNSSEC responses

DNSSEC responses are long, since they contain keys and signatures. Currently, 99.9% of the DNSSEC keys are of the RSA signature algorithm, using 1024 or 2048 bit keys and signatures [27]. This is unfortunate; it may have been better to use algorithms allowing shorter signatures and keys, reducing the length of DNSSEC responses and the associated abuses. In particular, it is estimated that comparable security can be achieved with much shorter keys and signatures, e.g., only 256 bit long, using elliptic curves cryptography; appropriate algorithms are standardised for DNSSEC, e.g., RFC 6605, [47].

However, currently merely 23 elliptic curve DNSSEC keys are in use (0.002%) [27]. This does not sufficiently motivate resolvers to support elliptic curves; hence, realistically, domains cannot use *only* elliptic curve signatures, since that is likely to be incompatible with most resolvers. Of course, domains could use both elliptic curve and RSA signatures. However, unfortunately, DNSSEC does not contain a 'ciphersuite negotiation' mechanism as in most other IETF cryptographic standards, e.g., TLS [48]. Hence, when a domain uses multiple algorithms (or keys), it will send *all* keys and all signatures to the client - increasing, rather then decreasing, the overhead.

The same problem may arise, when domains consider using longer RSA keys or a more secure algorithm, for improved security, or to change keys periodically. Domain are likely to

continue to use the old keys and algorithms as well for a long time or forever, due to the concern of interoperability with resolvers. All this implies obstacles to deployment of more appropriate cryptography, resulting in long DNSSEC responses and weak security. E.g., according to the experimental study carried out by SecSpider project, RSA1024-SHA-1 is the most supported option [27], although, 1024 byte RSA keys may not be sufficiently secure.

We recommend that the DNSSEC community consider developing appropriate, efficient mechanisms for cipher-suite negotiation, allowing domains to use multiple keys and algorithms while sending only these needed by the resolver. This may extend a related proposal, for signaling DNSSEC algorithm support [49].

VI. Conclusions

The DNS is a critical infrastructure component of the Internet, yet, the currently many DNS resolvers are vulnerable, in particular, to cache poisoning attacks. The DNSSEC standards [6], [7], [8], defined and evolved over more than a decade, extend DNS with cryptographic signatures, to provide security against Man-in-the-Middle attackers. DNSSEC deployment has progressed very slowly over the years, but gained some momentum recently with the signing of root and top level domains.

In this paper, we discussed several security and deployment problems with the current DNSSEC design. Several of these stem from the same problem essentially: DNSSEC requires transmission of non-standard DNS packets, in particular, significantly longer than standard DNS packets (limited to 512 bytes). The others are related to inter-domain dependencies and DNSSEC configurations. We outline the problems along with future directions to alleviate them.

References

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

[2] A. Herzberg and H. Shulman, "Security of patched DNS," in *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa,conf/esorics/HerzbergS12 Italy, September 10-12, 2012. Proceedings*, ser. Lecture Notes in Computer Science, 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

[3] ——, "Antidotes for DNS Poisoning by Off-Path Adversaries," in *ARES*. IEEE Computer Society, 2012, pp. 262–267.

[4] ——, "Fragmentation Considered Poisonous, or: One-Domain-to-Rule-Them-All.ORG, technical report 13-03," http://u.cs.biu.ac.il/~herzbea/security/13-03-frag.pdf, March 2013.

[5] A. Hubert and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers," RFC 5452 (Proposed Standard), Internet Engineering Task Force, Jan. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5452.txt

[6] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements," RFC 4033 (Proposed Standard), Internet Engineering Task Force, Mar. 2005, updated by RFCs 6014, 6840. [Online]. Available: http://www.ietf.org/rfc/rfc4033.txt

[7] ——, "Resource Records for the DNS Security Extensions," RFC 4034 (Proposed Standard), Internet Engineering Task Force, Mar. 2005, updated by RFCs 4470, 6014, 6840, 6944. [Online]. Available: http://www.ietf.org/rfc/rfc4034.txt

[8] ——, "Protocol Modifications for the DNS Security Extensions," RFC 4035 (Proposed Standard), Internet Engineering Task Force, Mar. 2005, updated by RFCs 4470, 6014, 6840. [Online]. Available: http://www.ietf.org/rfc/rfc4035.txt

[9] D. Dagon, M. Antonakakis, P. Vixie, T. Jinmei, and W. Lee, "Increased DNS forgery resistance through 0x20-bit encoding: security via leet queries," in *ACM Conference on Computer and Communications Security*, P. Ning, P. F. Syverson, and S. Jha, Eds. ACM, 2008, pp. 211–222. [Online]. Available: http://doi.acm.org/10.1145/1455770.1455798

[10] J. Bau and J. C. Mitchell, "A security evaluation of DNSSEC with NSEC3," in *Network and Distributed Systems Security (NDSS) Symposium*. The Internet Society, 2010. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/10/

[11] Alexa Web Information Company, "Top Sites," http://www.alexa.com/topsites, 2012.

[12] O. Gudmundsson and S. D. Crocker, "Observing DNSSEC Validation in the Wild," in *SATIN*, March 2011.

[13] D. York, "Challenges and opportunities in deploying dnssec," in *Securing and Trusting Internet Names (SATIN)*, March 2012.

[14] C. Partridge and G. Trewitt, "HEMS variable definitions," RFC 1024, Internet Engineering Task Force, Oct. 1987. [Online]. Available: http://www.ietf.org/rfc/rfc1024.txt

[15] J. Postel, "TCP and IP bake off," RFC 1025, Internet Engineering Task Force, Sep. 1987. [Online]. Available: http://www.ietf.org/rfc/rfc1025.txt

[16] P. Vixie, "Extension Mechanisms for DNS (EDNS0)," RFC 2671 (Proposed Standard), Internet Engineering Task Force, Aug. 1999, obsoleted by RFC 6891. [Online]. Available: http://www.ietf.org/rfc/rfc2671.txt

[17] B. Laurie, G. Sisson, R. Arends, and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence," RFC 5155 (Proposed Standard), Internet Engineering Task Force, Mar. 2008, updated by RFCs 6840, 6944. [Online]. Available: http://www.ietf.org/rfc/rfc5155.txt

[18] M. Larson, "DNSSEC Overview NANOG 51 Tutorial," NANOG 51 Tutorial.

[19] R. Bellis and L. Phifer, "DNSSEC Impact on Broadband Routers and Firewalls," Unpublished technical report, available at http://dnssec-deployment.com, September 2008.

[20] E. Osterweil, M. Ryan, D. Massey, and L. Zhang, "Quantifying the Operational Status of the DNSSEC Deployment," in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*. ACM, 2008, pp. 231–242.

[21] N. Weaver, C. Kreibich, B. Nechaev, and V. Paxson, "Implications of netalyzrs dns measurements," in *Proceedings of the First Workshop on Securing and Trusting Internet Names (SATIN), Teddington, United Kingdom*, 2011.

[22] Federal Executive Branch Internet Domains, "Listing of Federal Agency Internet Domains," http://explore.data.gov/Federal-Government-Finances-and-Employment/Federal-Executive-Branch-Internet-Domains/k9h8-e98h, February 2012.

[23] P. Mockapetris, "Domain names - implementation and specification," RFC 1035 (INTERNET STANDARD), Internet Engineering Task Force, Nov. 1987, updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604. [Online]. Available: http://www.ietf.org/rfc/rfc1035.txt

[24] E. Barker, W. Burr, A. Jones, T. Polk, S. Rose, Q. Dang, and M. Smid, "Recommendations for key management, part 3: Application-specific key management guidance, section 8: Dnssec," NIST, Tech. Rep. NIST Special Publication 800-57, Revision 3, December 2009.

[25] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendations for key management, part 1: General, table 4," NIST, Tech. Rep. NIST Special Publication 800-57, March 2007.

[26] P. Hoffman and W. Wijngaards, "Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC," RFC 6605 (Proposed Standard), Internet Engineering Task Force, Apr. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6605.txt

[27] E. Osterweil, "SecSpider deployment stats," http://secspider.cs.ucla.edu/stats.html, accessed May 2013.

[28] D. Conrad, "Indicating Resolver Support of DNSSEC," RFC 3225 (Proposed Standard), Internet Engineering Task Force, Dec. 2001, updated by RFCs 4033, 4034, 4035. [Online]. Available: http://www.ietf.org/rfc/rfc3225.txt

[29] S. Castro, M. Zhang, W. John, D. Wessels, and K. Claffy, "Understanding and preparing for dns evolution," *Traffic Monitoring and Analysis*, pp. 1–16, 2010.

[30] Alexa, "The web information company," http://www.alexa.com/.

[31] 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.

[32] H. Yang, E. Osterweil, D. Massey, S. Lu, and L. Zhang, "Deploying Cryptography in Internet-Scale Systems: A Case Study on DNSSEC," *Dependable and Secure Computing, IEEE Transactions on*, no. 99, pp. 1–1.

[33] S. Weiler, "DNSSEC Lookaside Validation (DLV)," RFC 5074 (Informational), Internet Engineering Task Force, Nov. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5074.txt

[34] D. J. Bernstein, "Breaking DNSSEC," 3rd USENIX Workshop on Offensive Technologies, August 2009.

[35] V. Ramasubramanian and E. Sirer, "Perils of transitive trust in the domain name system," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. USENIX Association, 2005, pp. 35–35.

[36] D. J. Bernstein, "DNS Forgery," Internet publication at http://cr.yp.to/djbdns/forgery.html, November 2002.

[37] Y. Gilad and A. Herzberg, "When Tolerance Becomes Weakness: The Case of Injection-Friendly Browsers," in *Proceedings of the International World Wide Web Conference*, May 2013.

[38] J. Lemon, "Resisting SYN Flood DoS Attacks with a SYN Cache," in *BSDCon*, S. J. Leffler, Ed. USENIX, 2002, pp. 89–97. [Online]. Available: http://www.usenix.org/publications/library/proceedings/bsdcon02/lemon.html

[39] R. Vaughn and G. Evron, "Dns amplification attacks," *Go online to http://www. isotf. org/news/DNS-Amplification-Attacks. pdf*, 2006.

[40] Arbor Networks. Worldwide infrastructure security reports series (2005-2011). http://www.arbornetworks.com/report.

[41] M. Prince, "The DDoS That Almost Broke the InternetThe DDoS That Almost Broke the Internet," April 2013.

[42] A. Studer and A. Perrig, "The Coremelt Attack," in *ESORICS*, ser. Lecture Notes in Computer Science, M. Backes and P. Ning, Eds., vol. 5789. Springer, 2009, pp. 37–52.

[43] F. Guo, J. Chen, and T. cker Chiueh, "Spoof detection for preventing dos attacks against dns servers," in *ICDCS*. IEEE Computer Society, 2006, p. 37.

[44] R. Tzakikario, D. Touitou, G. Pazi *et al.*, "Dns anti-spoofing using udp," Nov. 2009, uS Patent 7,620,733.

[45] "Open DNS resolver project," last retrieved May 2013. [Online]. Available: http://openresolverproject.org/

[46] D. Dagon, N. Provos, C. P. Lee, and W. Lee, "Corrupted DNS resolution paths: The rise of a malicious resolution authority," in *NDSS*. The Internet Society, 2008. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/08/papers/02_corrupted_dns_resolution.pdf

[47] P. Hoffman, "Elliptic curve digital signature algorithm (dsa) for dnssec," 2012.

[48] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," RFC 2246 (Proposed Standard), Internet Engineering Task Force, Jan. 1999, obsoleted by RFC 4346, updated by RFCs 3546, 5746, 6176. [Online]. Available: http://www.ietf.org/rfc/rfc2246.txt

[49] S. Crocker and S. Rose, "Signaling cryptographic algorithm understanding in dnssec," May 2013. [Online]. Available: http://tools.ietf.org/html/draft-ietf-dnsext-dnssec-algo-signal-10