4501 Network Security
Keaton Anderson
Jack Harner
Ben Hughes
Monique Mezher

DNSSEC

Introduction

The Domain Name System Security (DNSSEC) is a security extension of DNS which has protected users of the internet from malicious servers and kept user information private. In this paper, we will explore the topics of DNS, the history of DNSSEC's creation, the processes and architecture of DNSSEC, it and DNS vulnerabilities, and the alternatives posed to its use. We hope to guide others interested in DNSSEC to understanding its intricate workings, as well as contextualize and validate its place in the Internet world.

Overview of DNS

The Domain Name System (DNS) is a protocol and decentralized database involving servers around the world. It is responsible for the task of translating human readable web addresses (e.g. "example.com.") into Internet Protocol (IP) addresses (e.g. 152.35.231.1). This operates as a client-server model, where a client queries the DNS server for an IP address and the server either responds with a complete response; a reference to a server which may have the answer; or returns that the domain does not exist.This kind of request is also referred to as a forward DNS lookup. To ensure the communication is fast, DNS uses the User Datagram Protocol (UDP), with the tradeoff that no handshake is used to verify arrival.To elaborate more on this process, we must first look at how the DNS is structured.[11]

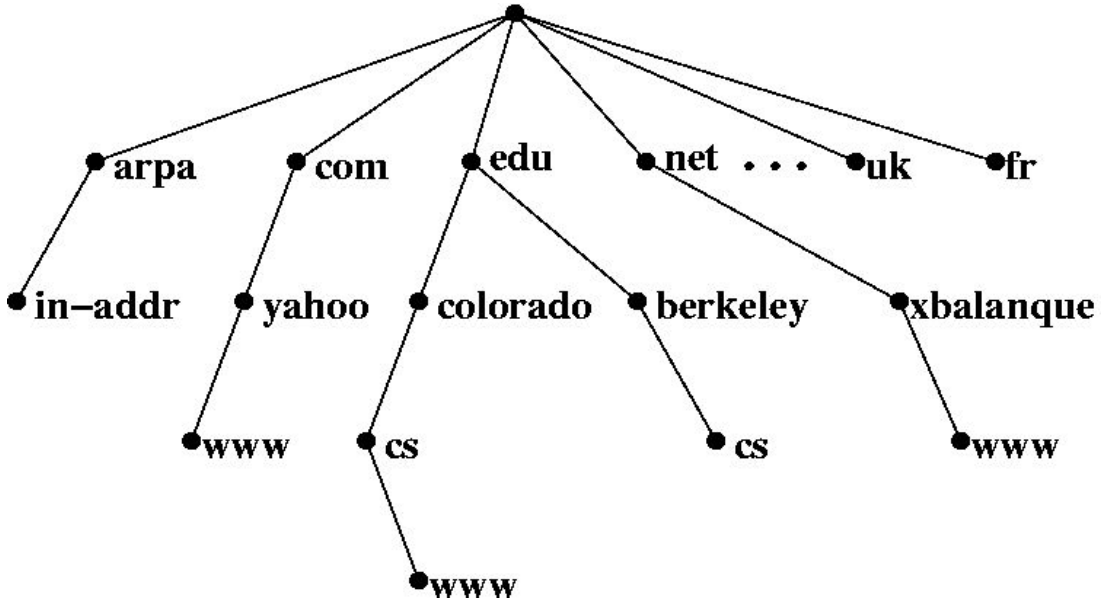

Figure A: Tree Structure of the DNS.
Source: http://ablogaboutnothinginparticular.com/?p=889

The DNS implements the domain name space which is laid out as a tree data structure, where each node is represented by a unique string called a domain name, such as “com.”. The domain of each node refers to it and all of its children. A zone refers to a node and its children which it is responsible for knowing everything about (at the very least, this must include its immediate children but in principle a parent could be responsible for the data of an entire sub-branch). The top node is called the root domain, and is indicated as a “.”. The domains one level below the root are referred to as Top Level Domains, such as the “com”, “edu”, and “biz” domains. The domain name is also referred to as a label, and a fully qualified domain name (FQDN) is created by joining a sequence of domain names by dots from right to left, each new level being a subdomain to the one before. We can imagine this sequence as traveling down the tree from the top or root node, indicated as a “.”, until we reached the domain name of specific (and unambiguous) host server. At this point we have a FQDN, such as “mail.google.com.” (note the dot at the end represents the root node). In reality a large set of servers may be responsible for the operation of its domain, and each node will have a set of authoritative name servers responsible for an official list of IP addresses and other data.[10]

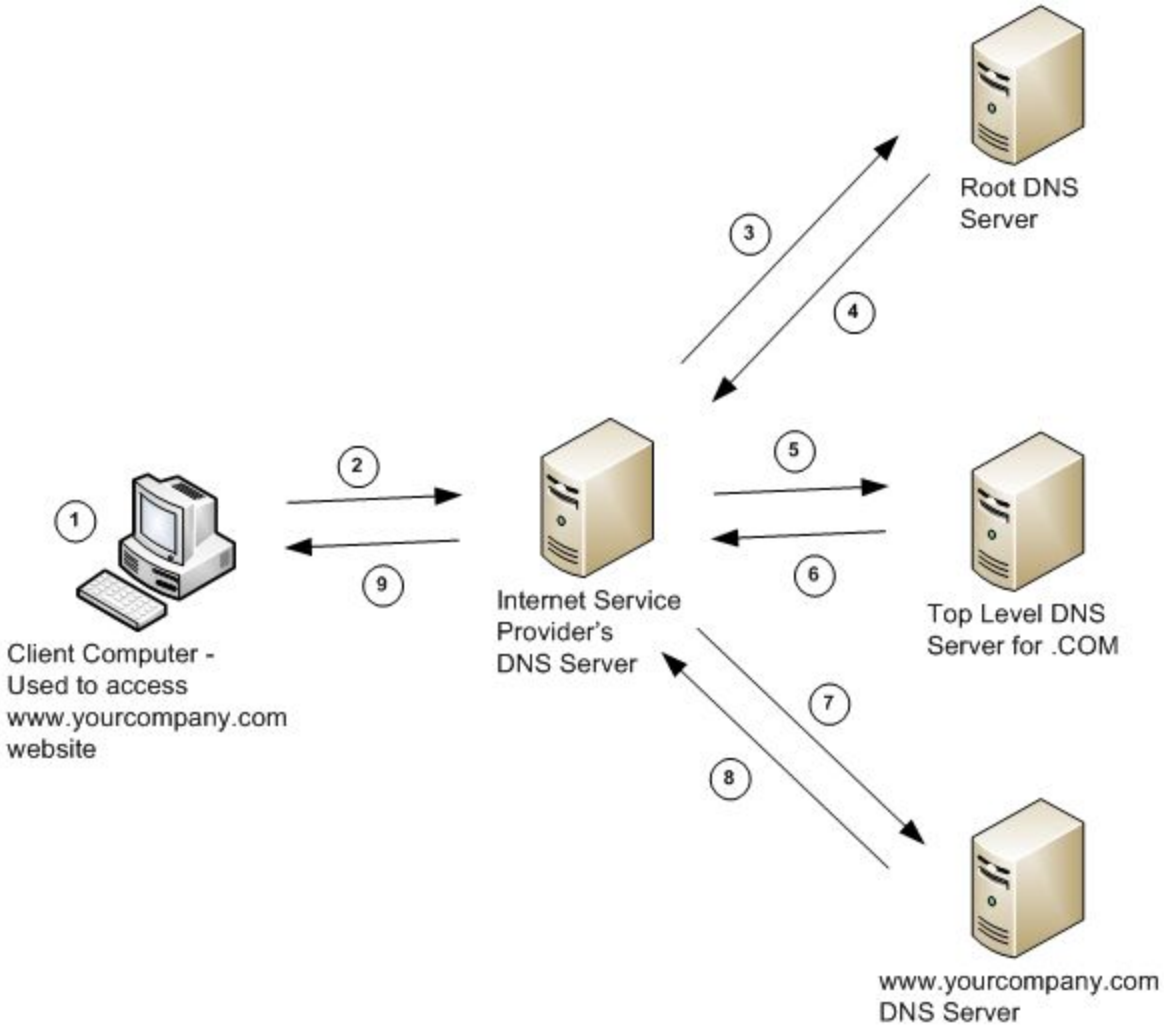

Figure B: How DNS works when not hacked
Source: http://www.uxworld.com/?p=384

When a client queries for an IP address this will either software on the client's device or a server provided by an Internet Service Provider will act as a DNS resolver. Often the requested IP will have been queried recently, and the DNS is made efficient through caching in its servers. If it is not found in cache the resolver may recursively query, making it fully responsible for returning an answer or error by querying other name servers as needed. Otherwise, it might iteratively query by sending non-recursive queries to chain of authoritative name servers, each one (not knowing the answer) responding with another name server lower in its domain. For example, if we wanted to go to collab.its.virginia.edu and our resolver did not have it cached, it may query a root server for answer. The root server, not knowing this IP address, would respond with the IP address of one of its edu servers, and so on, until collab.its.virginia.edu could respond with the IP address.[9]

Overview of DNSSEC

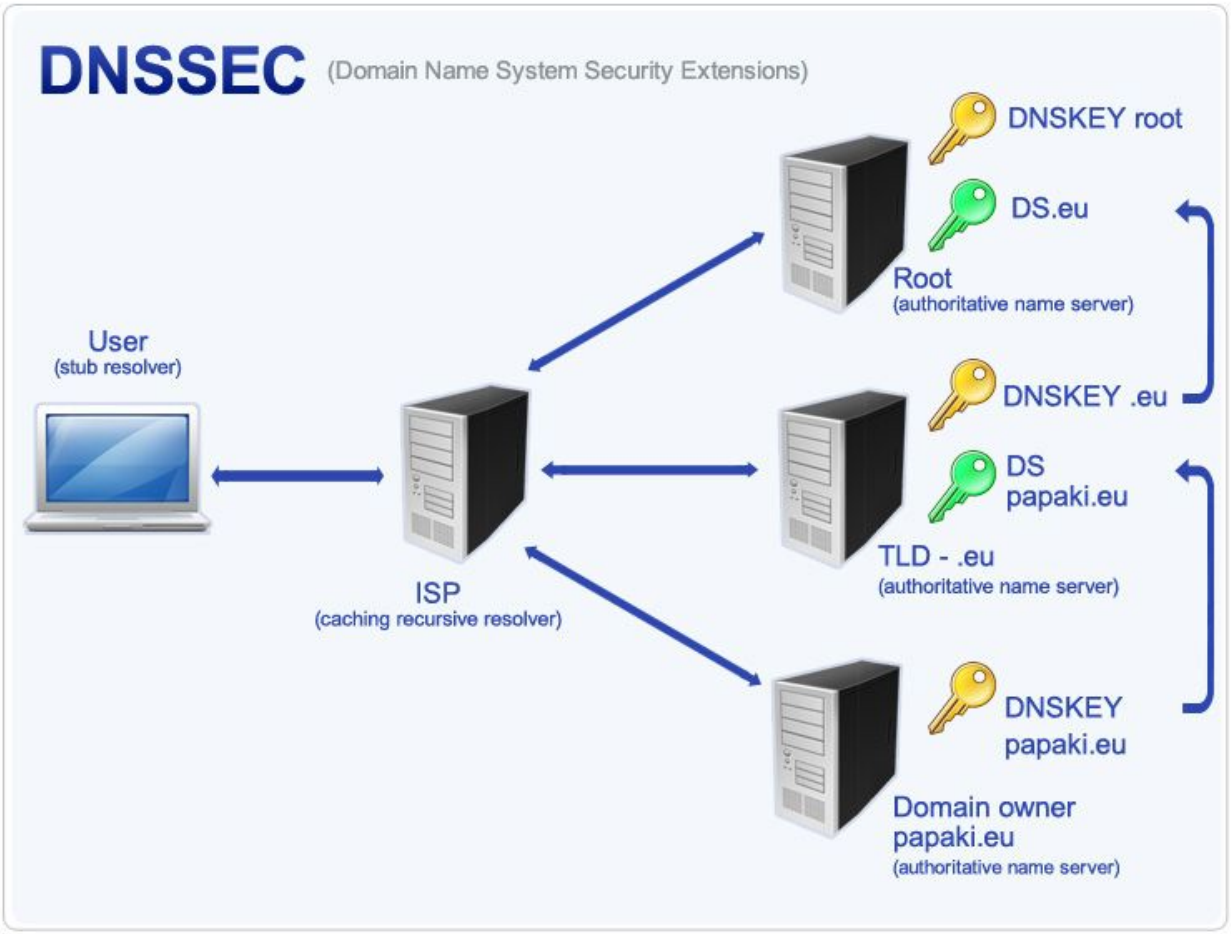

Figure C: An overview of DNSSEC, showing authoritative name servers, resolvers and keys. Source: https://cdn.papaki.com/sites/all/themes/papaki2/img/DNSSEC-graphix.jpg

DNSSEC originated from a fundamental lack of security in DNS, which soon led to abuse, most commonly cache poisoning. DNS needed a system that would verify the identify of the domain, allowing the user to avoid redirects and spoofing. The system introduced was known as DNSSEC, utilizing a suite of security records, resolvers and root servers inherent to DNS, to verify domain identities.

History of DNSSEC

The path to DNSSEC's creation began in the very beginning of DNS, where P.V. Mockapetris, the creator of DNS, discovered a vulnerability in his DNS architecture. However, this vulnerability was not addressed in a publication for over five years after its discovery.

Before DNSSEC's creation, the flaws in DNS had to be openly and publicly addressed to justify an increased security measure to users and Internet infrastructure moderators. In 1990, Dr. Steven M. Bellovin published a paper, *Using the Domain Name System for System Break-ins*, identifying vulnerabilities in DNS and exploiting them in order to show that DNS could be abused to reduce the system's security. This paper had been originally published to focus on the University of Berkeley's remote login being insecure, but proved relevant to any network services relying on host names for authentication.

This paper's publication spurred multiple other studies about DNS, including a new RFC addressing DNS's explicit vulnerabilities. RFC-2065, sending out the first call for comments and revisions on security extensions, first addressed the rationale of the publication, in that DNS had become an integral part of the Internet infrastructure but that its lack of security mechanisms was concerning to the point that extensions must be added to ensure continued use and success of use throughout the world wide web. These extensions, which would be the beginnings of DNSSEC, included the storage of authenticated public keys in the DNS, the zones in which the keys were authenticated, and key relations to DNS names. In addition, different kinds of keys and algorithms used for key generation were made compatible with DNS systems. Optional authentication of DNS protocol transactions was also implemented, but not used to its full extent during the time of RFC-2065's publication [21].

By 1999, the time of RFC-2535, *Domain Name System Security Extensions*, DNS's security extensions were a controversial standard that still required debate and tweaking through the Internet infrastructure community. It meant to standardize extensions of the DNS protocol to support widespread DNS security and public key distribution. The changes between RFC-2065 and RFC-2535 were marginal, but discrete; Extensions in RFC-2535 provided for the storage of authenticated public keys, and supported general public key services as well as aided DNS security by allowing resolvers to learn the keys authentication zones and the zones they were initially configured in. It also allowed for DNS requests to be authenticated.

These RFCs and the discussions that ensued because of them eventually led to the Internet Engineering Task Force (IETF) officially creating the Domain Name System Security Extensions (DNSSEC), adding data origin authentication and integrity to DNS. All of the extensions discussed in previous RFCs were standardized and strengthened. Their limitations were also published, in hopes for full transparency for what DNSSEC could provide. With this RFC, DNSSEC was officially ready for widespread Internet implementation [22].

DNSSEC's Architecture

Since DNS on its own is lacking defense features entirely, DNSSEC was implemented as a backwards compatible system that had to work with the existing DNS infrastructure. The primary intention was protecting the validity of DNS data from forgery or manipulation in transit so that domains are verified to be who they say they are. The security provided does NOT encrypt data nor does it protect data in transit like SSL. Instead it uses the tree structure of DNS to setup a system of zones and nodes that work through a chain of trust, in which child zones link to parent zones. This recursion continues up the tree to a root server, where the verification process begins and then continues down the zones again, as the root server of .com can verify its children zones, but not the children of its children nodes. This verification is the primary goal of DNSSEC and is done through DNS records, each with its own type and purpose, all grouped together and categorized into Resource Record sets (RRsets).[7]

Each RRset contains all the records of that type for each zone, and it is the full RRset that is signed by the root server, not individual records. There are several different types of records, the first that we will talk about is the *RRSIG*. The RRSIG record stores digital signatures for the entire recordset and is used by validators to approve RRsets. The record used to verify an RRSIG is the *DNSKEY*. These records hold the public key used in DNSSEC's public key cryptography and is assigned by the name authoritative server, and is then validated by the security enabled DNS resolver. This key can only be used to hold keys specifically for DNS signing. DNSKEYs have flags that signify its intended purpose, either as a Zone Signing Key (ZSK) or a Key Signing Key (KSK). KSK's are used to establish linkages between the parent and child nodes through keys, while the ZSK is used to sign all the RRsets in a zone. While the intended purposes for ZSK's and KSK's are very different, they both use public key cryptography to accomplish their purposes.[8]

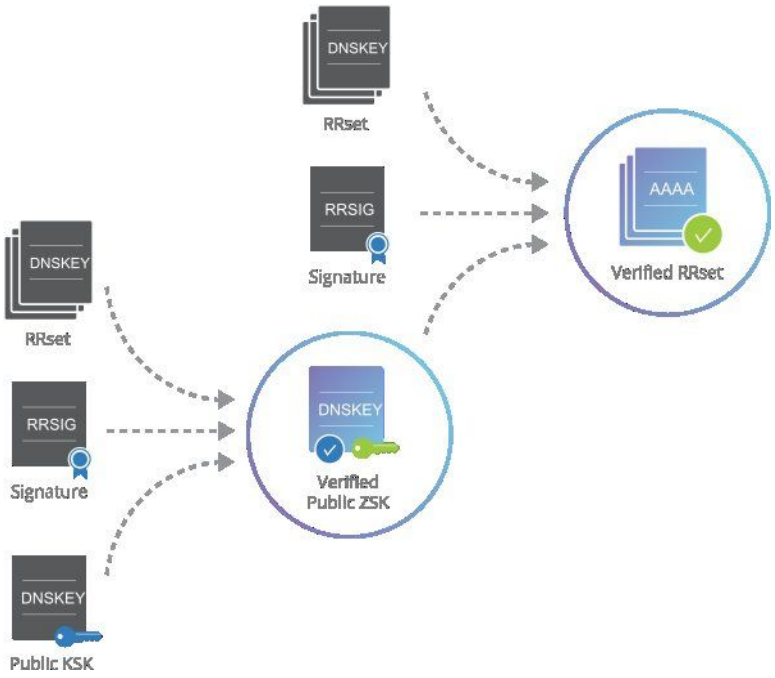

Figure D: An illustration of how signed RRsets function to verify DNSKEYs.
Source: cloudflare.com

While KSK's are used to create a link between child and parent zones, *NSEC3* records are important in creating links inside zones between record names. It contains hashed data listing the record types for the hash value in the label for the NSEC3 records name. Resolvers utilize these records to verify *denial of existence* for record names and types and are one protection against record spoofing.

The main record used in the establishment of the chain of trust between parent and child nodes is the *DS* record. The delegation signer (DS) record is a hashed DNSKEY record containing a child's public KSK which is placed in the parent zone. A child zone seeking to form a chain of trust with the parent will hash its own KSK to match with a parent zones DS, if the two are not the same then the bond is not formed. The DS record also contains a key tag which is used to help identify the referenced KSK for matching.[16]

The only required algorithm to implement for DNSSEC is RSA\SHA1- note the hash function is necessary to ensure integrity of the data. In practice RSA-NSEC3-SHAl is much more widely utilized, the difference being in how authoritative denial of existence is handled between NSEC and NSEC3, as we will explain in the next section. The recommended and most popular algorithm in use is RSA\SHA256, which we know still to be computationally secure. More recently ECDSA\SHA256 has seen wider use, an asymmetric cryptographic algorithm based off of elliptic curve cryptography which boasts smaller key sizes for similar levels of protection when compared to RSA [24].

How DNSSEC Works and Used Today

Before a chain of authentication can be established, the recursive name server or resolver must have a *trust anchor;* this is a public key stored in a DNSKEY, and as all public keys in the DNSSEC is either a ZSK or a KSK. The purpose of a KSK is to validate DNSKEY records in its respective zone (answering the question: *how do I know this is your public key?*) by using the private KSK to sign it and then providing the public KSK in another DNSKEY record.
The private ZSK signs all the other RRset records in the zone, and its public counterpart is made available in the DNSKEY signed by the KSK. In theory, a domain server could use the same key as its KSK and ZSK. In practice they are made different, due to the difficulty in *key rollovers*, a non-optional component of DNSSEC to routinely update keys.[13]

The above procedure still has a problem- how do we validate the DNSKEY record of the KSK? This is done with the DS record. When the parent zone is queried it provides in the response the DS record of the requested child, showing the child is DNSSEC-aware. When the resolver gets a response from the child server it compares a hash of the public KSK DNSKEY with the DS of its parent. A match authenticates the child's KSK, which can authenticate the child's ZSK and so on, enabling a chain of trust all the way down. Notice that the ZSK of a zone operates independently of its parent- a change simply means updating the RRsets and the encryption of the DNSKEY containing the new ZSK by the KSK. Updating KSK however requires updating the DS of the parent, and if configured incorrectly this rollover can disrupt all DNS queries in the zone. As a result the NIST recommends rolling over ZSKs about every 3 months, whereas KSKs can be rolled over every year.[12]

Given a trust anchor typically acquired outside the methods of DNS, we can use it as an entry point to verify other keys down the hierarchy. As of July 2010 the root node has been signed, greatly simplifying deployment by allowing the above procedure to permeate down the entire tree.[15]

The purpose of Next Secure (NSEC) and Next Secure 3 (NSEC3) records is to return a link to the 'closest' record name of the requested domain. These have their own corresponding RRSIG records so a response can be authenticated with same procedure with the last zone's ZSK. The potential issue raised by NSEC records is that they can be used to 'zone walk,' a procedure by which an adversary can enumerate all the domain names in a zone by issuing phony requests to non-existent domains. Since the zones are sorted if a query for "store.google.com" returns in an NSEC "mail.google.com" we now know there are no records for domains between "store." and "mail." in the google domain. NSEC3, as mentioned in the architecture section, uses a sorted list of *hashes* of the zones, and tries to match the hash of the relevant query data with them. This makes enumeration of the domain more difficult, as an attacker will either need to use a brute-force method- which may be detected- or rainbow tables of hashes of potential domain names in the zone. This can be made even more computationally secure by adding salts to the domain names.[14] The NSEC3PARAM lets an authoritative name

server know what these parameters are, such as the hash algorithm, salt parameters, and how many times to use the hash function.[17]

In order to manage the increased packet size from adding these extra records servers must have Extensions Mechanisms for DNS (EDNS0) enabled, laid out in RFC 2671. This adds extra flag bits which facilitate DNSSEC communications. When a resolver wants to do DNSSEC on behalf of a client it sets the DO ("DNSSEC OK") flag, letting the queried DNS servers know to send back DNSSEC data. A CO ("Checking Disabled") flag determines if a response is returned even if it is not validated and will be set if the resolver is capable of DNSSEC validation. If an authoritative server is queried it will set the AA ("authoritative answer") bit which can be verified by the resolver. If a query was for an authoritative server this can take over the roll of the AD ("authenticated data") flag, which tells the client in general whether the answer was DNSSEC validated.[16]

There are several different metrics to measure the adoption rate of DNSSEC. On the one hand, at about 90% most of the TLDs have been signed by the root and are capable of DNSSEC. The percentage however drops drastically just one level down the tree, with only an overall 4% DNSSEC-capable. This varies significantly by domain- .gov is about 90% signed whereas .com is 0.5% signed. It is estimated that about 13% of users validate exclusively through DNSSEC, which has remained relatively stable for the past few years.[17] About 80% of users request that their response to DNSSEC validated. This also varies regionally- Sweden is validating more than 75% of its users' queries, whereas the UK validates less than 5%.[19]

There are several reasons for the delay in adoption. Most ISPs (except, notably, Comcast) do not have DNSSEC-aware resolvers. They may sign contracts with media companies which require specific kinds of DNS load balancing, where client queries are strategically distributed to different servers for the same domain to maximise user performance. An ISP which enables DNSSEC may also receive the complaints because of a misconfiguration in a domain- this happened in 2012 when NASA made an error in a key rollover and Comcast's users were denied access to the site. Many registrars for domain names have not enabled DNSSEC as they would bear a much larger administrative burden- ICANN requires they be able to alter public key material for those domains which need it. Lastly, there still exist misunderstandings of what DNSSEC provides and how it is different from other security measures (like SSL\TLS).[20]

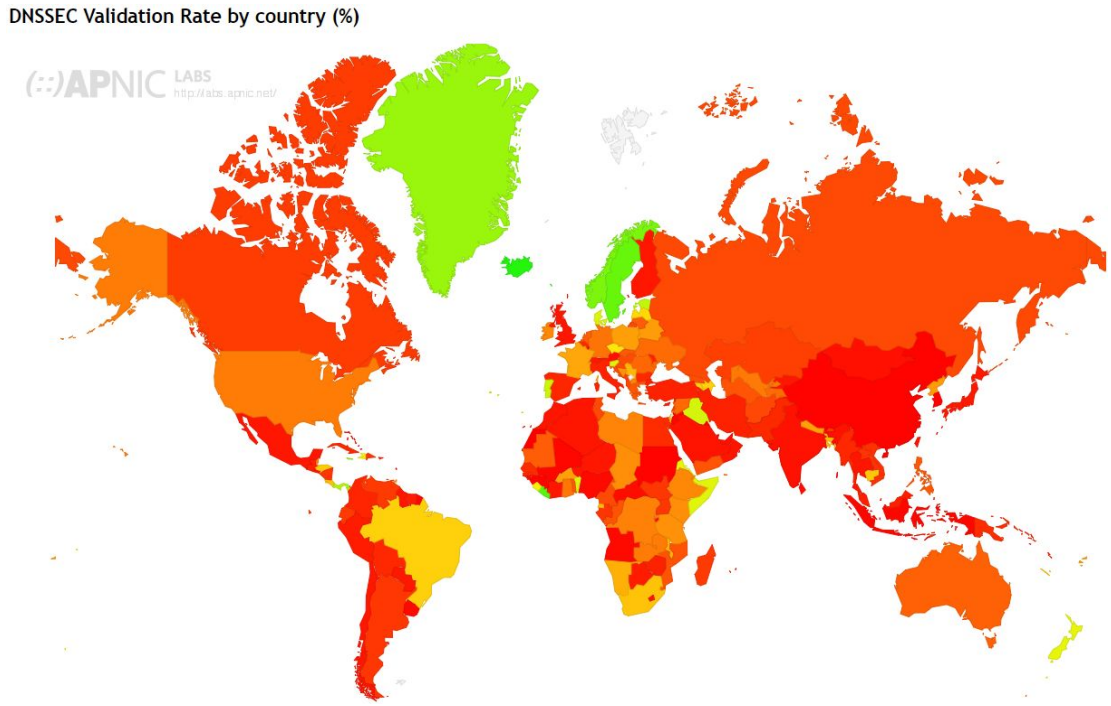

Figure E: DNSSEC Validation Rate by Country. Green: > 60%, Red: < 10%.
Interactive Source: https://www.youtube.com/watch?v=VaNi34cn9uE

Vulnerabilities of DNSSEC: DNS/DNSSEC Amplification Attacks

A DNS amplification attack is when an attacker uses something relating to the domain name system to amplify the effect of a DDOS attack.

A normal response from DNS over UDP was fixed at 512 bytes. For practical purposes the maximum DNSSEC response is 4 KB, but DNS servers typically set it to 1472 or 1232. This response is much larger than a normal DNS query, and so attackers can use this to amplify the effect of their DDOS attack. In a DNSSEC amplification attack, the attacker sends a request to a server using DNSSEC with the "ANY" query (returns every DNS record that exists for a name, and is designed for debugging) while using a spoofed IP address, the spoofed IP address being the target of the attack. The DNS server then responds to the request and sends the large response that is the DNSSEC response to the spoofed IP address. Using a distributed botnet this can be many times the communication an IP address is used to handling, and it will not work properly. This means that DNSSEC can be used by an attacker to harm another party. Also, if the domain owner pays for DNS by the query, this can significantly increase their DNS bill.

Although this can cause a problem it is not as significant as some people make it out to be. Attackers wanting to make a DNS amplification attack have several more effective tools they can use, so using DNSSEC for this purpose isn't their best option. Some of the other amplifiers attackers can use for a DDOS attack other than DNS are: SSDP, SNMP, NTP, and any TCP

server in some cases. The real problem lies with systems that generate traffic with spoofed IP addresses and networks that allow it.
A study done by Neustar, as documented in "DNSSEC: How Savvy Attackers Are Using Our Defenses Against Us", it was found that the average amplification of a DNSSEC amplification attack to be 28.9 Gbps. Neustar recommends that organizations that use DNSSEC not respond to the "ANY" query.

Vulnerabilities of DNSSEC: Other DNSSEC Vulnerabilities

Most of the vulnerabilities of DNSSEC revolve around DNSSEC not being implemented correctly. While it can be said that people should implement it correctly, the problem lies in that it is difficult to implement because of its complexity, so realistically this is still a major problem. In implementing DNSSEC careful security considerations should be applied to the keys. The size, algorithm, and validity period are important. The longer that a key is in use, the greater the chance that it may have been compromised. Key rollover needs to be done carefully. The zone private key should not be in server memory, and should exist in a secure physical place. Root key rollover is important and has an impact on the whole DNSSEC structure.

Famous DNS Exploit: The Kaminsky Bug

Perhaps the most famous vulnerability of DNS, leading to the mass adoption of DNSSEC, was in 2008, when Dan Kaminsky presented a vulnerability within DNS' framework which allowed for arbitrary, immediate DNS cache poisoning.

Kaminsky's paper describes a scenario in which an attacker is able to set up his own nameserver to work as a faux authoritative zone for any site they wished. Due to DNS being set up in such a way that local caches in recursive nameservers are thoroughly checked to be a quicker and more efficient Content Delivery Network (CDN), this attacker, once checked once for a specific website validation, will now be saved to the local cache and be used regularly to fulfill future queries of that specific site.

The reason which this cache poisoning is extremely dangerous is the attacker's ability to redirect users to malicious sites, while not informing the user that the site is not the original site they requested. Without DNSSEC being able to identify and confirm the nameserver and authoritative zones are authentic, DNS alone lacked the authority to tell users whether or not the website they were directed to was confirmed legitimate through a good nameserver. These malicious sites, which mimicked the genuine sites users intended to be directed to, would then be able to steal any information the user presented to it, as well as send to the user malware and spyware if the user tried to download data from that site.

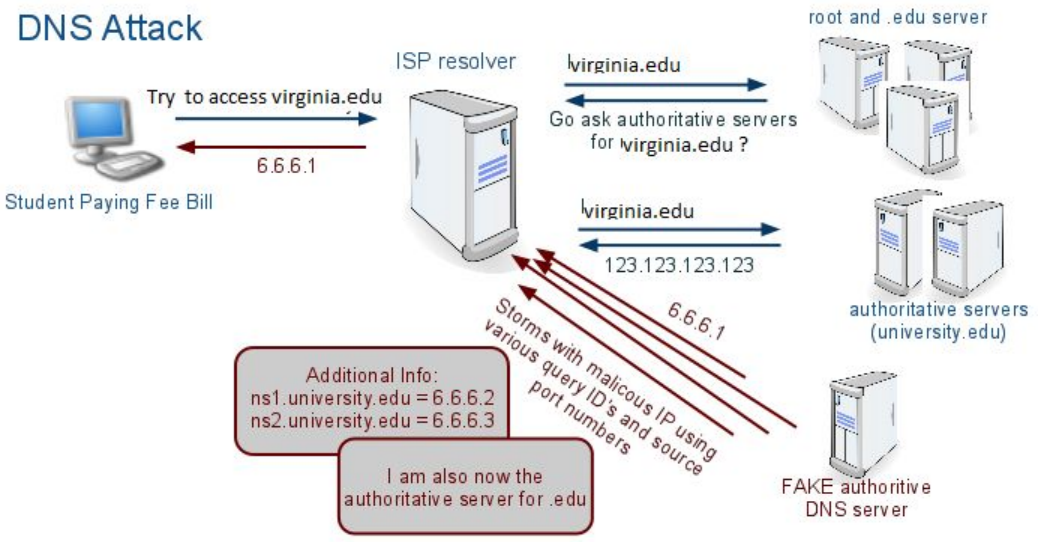

Figure F: DNS Attack with Kaminsky Bug Principles. Source: er.educause.edu

The detailed process of attacking DNS using the Kaminsky bug process is outlined in Figure B above. When a user tries to access a specific website, a fake authoritative DNS server is able to spam the ISP resolver with query IDs and fake, malicious IPs. If the server does this step enough times to overwhelm the resolver and overtake the real authoritative and .edu servers, then the fake DNS server will be considered the true server of interest for the student. This will thus be the server that the user is directed to, where they will then be exposing their information to an insecure, malicious server that is able to store and compromise their personal information. Anyone who used this website, or was targeted by the fake authoritative DNS server, would then be in danger of identity theft and destruction of their privacy [23].

However, the bug which Kaminsky showed was not originally taken seriously. Although DNS vendors and founders discussed the bug shortly after its exposure, their solution originally implemented only delayed the attack that had originally taken 10 seconds to take hours or days. Getting rid of the immediate attack was not enough, however, and the Internet Engineering Task Force (IETF) began drafting and addressing the vulnerabilities exposed by Kaminsky, as well as others speculated by other researchers, to create the extensions we are familiar with as DNSSEC today. Therefore, DNSSEC would not be here today without this bug exposure- it was a direct reply to Kaminsky's research.

Alternatives to DNSSEC

Although DNSSEC has provided reliable, secure authentication of domains and subdomains, it isn't perfect. Alternative security measures have been implemented and tested with DNS, with similar use cases. We will go into detail about two of the main alternatives for heightened security: DNSCurve and TSIG.

DNSCurve is a protocol extension that adds link-level security to DNS, meaning that per connection, DNSCurve checks the authenticity and reliability of the server software and registry services it is presented with, as well as cryptographically secures packets of clients. DNSCurve uses two different packet formats for distinct case scenarios: a predominantly backwards-compatible format to use with DNS proxies and firewalls, and a streamlined packet format that takes up little space and is thus useful for quicker, more secure protocols. It is a cryptosystem, implementing a combination of three different cryptographic message authentication codes- Curve 25519, Salsa20, and Poly1305- in order to do its message authentication. The cryptosystem itself relies on 256-bit public and secret keys, 192-bit nonces, and 128-bit authenticators.[1]

DNSCurve keys are distributed differently for clients and servers. For servers, public keys are distributed by encoding via the server names embedded in DNS NS records. Clients, on the other hand, distribute their respective public keys in their query packets. However, an extra layer of security is then made possible if a client knows the DNSCurve public key for the server; they are able to use that public key with the client's secret key and a nonce to hide the query after encryption. This provides a useful, not DNSSEC-possible trick for keeping confidentiality of information.

In essence, DNSCurve, when implemented on server and client side networks, acts as a cryptographic protection for client queries and packets. Attackers, usually gathering information from packets and DNS queries, would be unable to understand information from the packets, and would not be able to do any kind of network/IP spoofing. However, it is not a perfect construction of a more secure DNS setup than DNSSEC. In order for DNSCurve to be a reliable source of security, all servers must be set up and recognized by DNSCurve- without doing so, the cryptographic security is diminished entirely, as no encryption of the packets will take place.[4]

However, supporters of DNSCurve are quick to point out that DNSSEC has times of outages which cause both inconvenience to users and panic to domains, especially when root servers time out, whereas DNSCurve would constantly be turned on and not necessary to be managed after.[2] The main advantage to DNSSEC that is noted is that DNSSEC could, in theory, protect DNS data if the administrator generates keys and signatures on a separate, non-compromised computer, making the attacker who compromises domain name servers essentially stranded. However, supporters note this does not often happen, and without DNSSEC being encrypted, information can leak easily.[3]

Another alternative, allowing for transaction level authentication using shared secrets and one way hashes, is *Transaction Signature* (TSIG). Its original intent was to be used to authenticate dynamically whether network traffic (dynamic update requests) comes from approved clients or approved name servers recursively. It takes off of the fact that DNS relies on caching DNS servers on other hosts in order to authenticate requests, and thus uses a static authentication rather than a dynamic approach usually. It also takes advantage of secret key

based MACs to authenticate DNS update requests as well as transaction responses in a simplistic, lightweight form- getting rid of heavy overhead setup.[5]

However, another major issue under the same precursor is that the secret keys must be distributed securely, which has proved unreasonable to assume proper care of in the past. Trust relationships, and the sharing of keys, are impractical to rely on, particularly in a highly connected network. The TSIG proposal, therefore, is disregarded for server to server authentication or any kind of connection with many to many relations with other clients or servers. Not only this, but the original hash used for signing- the MD5 hash- has been theoretically, practically, and physically broken as use of a cryptographic hash function. Any protocols using it, therefore, should also be ceased for mass use.

There have been standard cases of use for such a protocol, however, such as local area networks with multiple internal servers where key distribution can happen via a third party or physically between masses. In this case, the TSIG assures that DNS information from certain servers is from who they say they are, which keeps authenticity of the server with a lightweight, non standardized fashion. It also assures no reliance on root servers that are not in one's own network.[6] This is to prove that although DNSSEC has found use cases in the massive Internet networking system, it is not the only nor most viable solution for other use cases.

Conclusion

In this paper, we have discussed the history, rationale, and exploits of DNS and the creation, observation, and intricacies of DNSSEC. We have touched upon the context and concerns which led to DNSSEC's creation from the Kaminsky Bug, as well as discussed alternatives, such as DNS Curve and TSIG. Regardless of the future findings of security flaws using DNSSEC, it is undeniable that it has been an integral part of the Internet's history and will continue to secure millions of users transactions, private information, and connections with other websites for the outstanding future- unless there is another unforeseeable crack in this secure protocol design.

References

1. Demsky, M. *DNSCurve: Link-Level Security for the Domain Name System.* IETF Tools, IETF, 26 Feb. 2010, tools.ietf.org/html/draft-dempsky-dnscurve-01
2. *Major DNSSEC Outages and Validation Failures*, Ianix, ianix.com/pub/dnssec-outages.html.
3. *Differences between DNSCurve and DNSSEC*, dnscurve.io/faq/differences-between-dnscurve-and-dnssec.html.
4. *DNSCurve: Usable Security for DNS*. 26 June 2009, dnscurve.org/in-benefits.html.
5. Vixie, P., et al. *Secret Key Transaction Authentication for DNS (TSIG).* IETF, May 2000, www.ietf.org/rfc/rfc2845.txt.
6. Raftery, James. *Securing Your DNS Information with Transaction Signatures (TSIG)*. 17 Jan. 2001,romana.now.ie/james/tsig.html.
7. Arends, R., et al. *Resource Records for the DNS Security Extensions.* IETF, Mar. 2005, tools.ietf.org/html/rfc4034.
8. "How DNSSEC Works." Cloudflare.com, Cloudflare, Inc, www.cloudflare.com/dns/dnssec/how-dnssec-works/.
9. "DNS Architecture." Microsoft.com, technet.microsoft.com/en-us/library/dd197427(v=ws.10).aspx.
10. Mockapetris, P. *Domain Names - Concepts and Facilities.* IETF, https://tools.ietf.org/html/rfc1034
11. Karrenberg, Daniel. *The Internet Domain Name System Explained for Non-Experts*. The Internet Society, 1 Mar. 2004, www.internetsociety.org/wp-content/uploads/ 2017/09/ The-Internet-Domain-Name-System-Explained-for-Non-Experts-ENGLISH.pdf.
12. Karrenberg, Daniel. "NANOG 67." Internet Systems Consortium, *DNSSEC Tutorial*, 1 Mar. 2004, www.isc.org/wp-content/uploads/ 2016/06/Winstead_DNSSEC-Tutorial.pdf.
13. Arends, R., et al. *DNS Security Introduction and Requirements*. IETF, Mar. 2005, tools.ietf.org/html/rfc4033.
14. Strotmann, Carsten. *Take Your DNSSEC with a Grain of Salt*. 1 Dec. 2012, info.menandmice.com/blog/bid/73645/Take-your-DNSSEC-with-a-grain-of-salt.
15. "The DNSSEC Root Signing Ceremony." *Cloudflare.com*, Cloudflare, Inc, www.cloudflare.com/dns/dnssec/root-signing-ceremony/.
16. "Overview of DNSSEC." *Microsoft.com*, 11 Feb. 2014, technet.microsoft.com/en-us/library/jj200221(v=ws.11).aspx.
17. Laurie, B., et al. *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*. IETF, Mar. 2008, tools.ietf.org/html/rfc5155.
18. York, Dan. *State of DNSSEC Deployment*. The Internet Society, 13 Nov. 2016,

iepg.org/2016-11-13-ietf97/IEPG-DNSSEC-Deployment-1.pdf.

19. *Use of DNSSEC Validation for World (XA)*. APNIC, stats.labs.apnic.net/dnssec/XA?c=XA&x=1&g=1&r=1&w=7&g=0.

20. York, Dan. *Challenges and Opportunities in Deploying DNSSEC*. The Internet Society. 23 Mar. 2012, www.internetsociety.org/wp-content/uploads/2012/03/SATIN2012-DNSSEC-Challenges-v12-FINAL.pdf.

21. "A short history of DNSSEC." LNet Labs, 25 Sept. 2013, www.nlnetlabs.nl/projects/dnssec/history.html.

22. Crocker, Steve. *DNSSEC History, Status, Future.* Internet Dagarna 2008, 21 Oct. 2008, internetdagarna.se/arkiv/2008/www.internetdagarna.se/images/stories/pdf/domannamn/Steve_Crocker_administrationofdnssec.pdf.

23. Friedl, Steve. *An Illustrated Guide to the Kaminsky DNS Vulnerability.* 08 Aug. 2018, http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

24. *Global DNSSEC Deployment Tracking*. Verisign, secspider.verisignlabs.com/stats.html.

26. "DNSSEC: How Savvy Attackers Are Using Our Defences Against Us" *Neustar* April 2016, https://ns-cdn.neustar.biz/creative_services/biz/neustar/www/resources/whitepapers/it-security/d ns/neustar-dnssec-report.pdf

27. "DNSSEC and DNS Amplification Attacks" *Microsoft* April 23, 2012, https://technet.microsoft.com/en-us/security/hh972393.aspx

28. "DNS Amplification Attacks" *US-CERT* March 29, 2013 https://www.us-cert.gov/ncas/alerts/TA13-088A

29. "Security Vulnerabilities in DNS and DNSSEC" *Suranjith Ariyapperuma and Chris J. Mitchell* April 2007, http://web.mit.edu/6.033/www/papers/dnssec.pdf

30. "A Security Evaluation of DNSSEC with NSEC3" *Jason Bau and John C. Mitchell* January 2010, https://eprint.iacr.org/2010/115.pdf