SIDN Labs

https://sidnlabs.nl

April 23, 2022

Author Version

Title: Addressing the challenges of modern DNS: a comprehensive tutorial

Authors: Olivier van der Toorn, Moritz Müller, Sara Dickinson, Cristian Hesselman, Anna Sperotto and Roland van Rijswijk-Deij

Published in: Computer Science Review

DOI: https://doi.org/10.1016/j.cosrev.2022.100469

Addressing the Challenges of Modern DNS
A Comprehensive Tutorial

Olivier van der Toorn[a,*], Moritz Müller[a,b], Sara Dickinson[c], Cristian Hesselman[a,b], Anna Sperotto[a], Roland van Rijswijk-Deij[a,d]

[a] *University of Twente, PO Box 217, 7500 AE Enschede, The Netherlands*
[b] *SIDN Labs, PO Box 5022, 6802 EA Arnhem, The Netherlands*
[c] *Sinodun Internet Technologies Ltd., Robert Robinson Avenue, Oxford OX4 4GA, United Kingdom*
[d] *NLnet Labs, Science Park 400, 1098 XH Amsterdam, The Netherlands*

Abstract

The Domain Name System (DNS) plays a crucial role in connecting services and users on the Internet. Since its first specification, DNS has been extended in numerous documents to keep it fit for today's challenges and demands. And these challenges are many. Revelations of snooping on DNS traffic led to changes to guarantee confidentiality of DNS queries. Attacks to forge DNS traffic led to changes to shore up the integrity of the DNS. Finally, denial-of-service attack on DNS operations have led to new DNS operations architectures. All of these developments make DNS a highly interesting, but also highly challenging research topic. This tutorial – aimed at graduate students and early-career researchers – provides a overview of the modern DNS, its ongoing development and its open challenges. This tutorial has four major contributions. We first provide a comprehensive overview of the DNS protocol. Then, we explain how DNS is deployed in practice. This lays the foundation for the third contribution: a review of the biggest challenges the modern DNS faces today and how they can be addressed. These challenges are (i) protecting the confidentiality and (ii) guaranteeing the integrity of the information provided in the DNS, (iii) ensuring the availability of the DNS infrastructure, and (iv) detecting and preventing attacks that make use of the DNS. Last, we discuss which challenges remain open, pointing the reader towards new research areas.

Keywords: DNS, DNSSEC, security, availability, Internet abuse

1. Introduction

The Domain Name System (DNS) is *the* naming system of the Internet. In its most basic form, it translates human readable domains names into Internet Protocol (IP) addresses. For example, the domain `example.com` is translated to `93.184.216.34`. Typically, every time a client wants to connect to a server via a domain name, this name needs to be translated to an IP address. If the DNS query fails, the server, despite being online, becomes unreachable to the client.

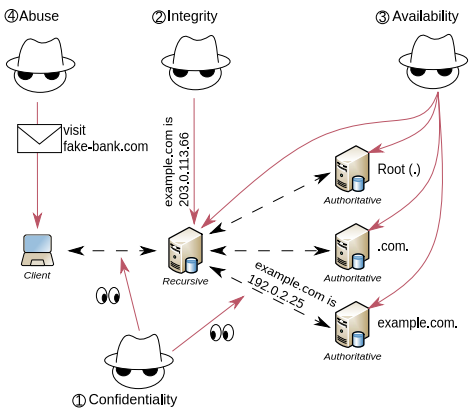


Figure 1: Challenges in the DNS

*Corresponding author
Email addresses: `o.i.vandertoorn@utwente.nl` (Olivier van der Toorn), `moritz.muller@sidn.nl` (Moritz Müller), `sara@sinodun.com` (Sara Dickinson), `cristian.hesselman@sidn.nl` (Cristian Hesselman), `a.sperotto@utwente.nl` (Anna Sperotto), `r.m.vanrijswijk@utwente.nl` (Roland van Rijswijk-Deij)

Table 1: Document structure

Part 1: Background

Section 2 Core Concepts	*Section 3 DNS Evolution*	*Section 4 Measuring the DNS*
The Origins of the DNS	Resolvers	Passive measurements
Message format, components and actors	Authoritative name servers	Active measurements
	Modern DNS	Comparison

Part 2: Challenges				**Part 3: Alternatives**
Section 5 Confidentiality	*Section 6 Integrity*	*Section 7 Availability*	*Section 8 Abuse*	*Section 9 Alternatives*
On-the-wire	History of DNSSEC	Capacity	Attack facilitating	RAINS (SCION)
Data in DNS Servers	Larger messages	Multiple servers	Communication	Named Data Networking
In practice	DNSSEC Signing	Anycast	Attack excerbating	Block-Chain based
Encryption and DNSSEC	DNSSEC Validation	Monitoring		Open Challenges
	Deployment	Caching		
	Zone files	Hardening		

The specifications that today's DNS is based on date back to 1987. Back then, the designers could not have foreseen the scale at which the DNS would be deployed[1]. Regardless of the original designers' intentions, the DNS can be considered a major success. Today, the DNS is as important as ever with billions of physical devices connected to the Internet, relying on a functioning DNS [3]. In the late eighties the Internet was still a safe place where users trusted each other, intentional attacks on Internet infrastructure did not occur and privacy was not a concern either. This has changed considerably over the past 30 years and has put increasing pressure on the DNS. DNS queries became of interest for Internet Service Provider (ISP) that use them to learn more about their customers [4] (① in Figure 1), attacks are launched to tamper with the information in the DNS to direct users to malicious content [5] ②, and the infrastructure that runs the DNS is constantly undergoing denial-of-service attacks, threatening its availability [6] ③. At the same time the success of the DNS makes it attractive for offenders by unwillingly helping them to enable, manage, and fuel attacks, e.g. in order to direct end-users to malicious websites ④.

We summarize these pressures on the DNS in *four main challenges*: (i) *confidentiality* of DNS queries, (ii) *integrity* of information stored and sent in the DNS, (iii) *availability* of the underlying DNS infrastructure, and (iv) *abuse* of the DNS in attacks and distribution of harmful content on the Internet. Over time, multiple extensions and tools have been developed to address these challenges, contributing to the ongoing success of the DNS. Despite this effort, not all challenges were addressed and some in the Internet Engineering Task Force (IETF) have even raised the question if it may be time for a major rethinking of the DNS [7].

The goal of this tutorial is to provide the reader with comprehensive knowledge on *challenges* and their *solutions* in the *modern* DNS. As we will discuss in the next section, the total corpus of specifications on the DNS now exceeds 3,500 pages. This makes it exceedingly hard for researchers and practitioners to understand the intricacies of today's DNS with all its challenges. Therefore, in this paper we will:

i) Provide a thorough explanation of the DNS protocol, going beyond basic tutorials, which lays the foundation for understanding the challenges and solutions.

ii) Give an overview of how DNS is deployed and explain what changed over time.

iii) Cover all major challenges for the DNS, their current solutions, and illustrate challenges and research directions.

iv) Discuss open challenges that still need to be addressed, looking at non-DNS naming systems for inspiration.

[1]Interviews on the origin and adoption of the DNS with Paul Mockapetris [1] and Paul Vixie [2] aptly illustrate this.

Objective and Approach. The target audience of this article are graduate students and early-career researchers with an interest in the DNS. Following the explanation of the basics of the DNS, we discuss existing and new *real world* challenges of modern DNS, targeting readers familiar with the aforementioned basics. We hope our paper equips researchers with the knowledge necessary to discover new fields of research and develop or improve solutions to presented challenges in this paper. As a starting point, we touch on some open challenges in this article.

The information in this article stems from multiple sources. Functions and architecture of the DNS are defined in numerous Request for Comments (RFC) standards of the IETF and we refer to the most relevant ones throughout the article. Research on the security aspects of the DNS dates back to the 1990s [8]. We combine the most relevant information from academic papers, research, feedback from operators of large DNS services, and the author's own experiences [9, 10, 11, 12] and expertise of the DNS, supported by our own DNS measurements to provide a current guidebook on security and resilience of the DNS.

Comparison with other DNS Surveys and Tutorials. There exist nearly 200 standardization documents (RFCs) [13], numerous books (e.g. [14, 15, 16]), and countless tutorials online describing and explaining the DNS.

Kim et al. [17] provide a more high level survey on DNS security, discussing threats and mitigation strategies. Also, Khormali et al. [18] carry out a survey which focuses mostly on the aspects of DNS security, touches on the issue of integrity, confidentiality and DNS measurements, and provide additional insights into machine learning algorithms used for DNS analysis. In comparison, in this article we give readers, new to the DNS, more hands-on knowledge to carry out their own research. Chandraramouli et al. [19] discuss the challenge of integrity in the DNS, but their article is more than 14 years old. As this article, the survey paper by Zou et al. [20] discusses some alternative naming systems as well, but not other security challenges. Other survey papers on DNS security exist, but leave out many details about underlying issues and solutions [21, 22]. Also, surveys have been published that focus on some aspects of our article, e.g. botnets [23], malicious domain names [24], the detection abuse in the DNS [25], or phishing [26]. To the best of our knowledge, no article exists which integrally describes the essential aspects and the modern challenges of the DNS and how to address them.

Reading Guide. This tutorial is divided into three main parts, each divided into multiple sections. Table 1 provides an overview of all the parts and sections. Part 1 covers the background of the DNS and equips the reader with the necessary knowledge to understand the challenges and solutions and to do independent research. It includes a discussion of the origins of the DNS and introduces core concepts and techniques to measure the DNS. In this part, we also explain how DNS deployments have evolved. Part 2 discusses the four main challenges: confidentiality, integrity and availability of the information in the DNS and the misuse of the DNS, each in separate sections. Each section ends with an overview of the challenges remaining. Finally, in Part 3, we describe how other naming systems attempt to address these challenges and we discuss if their solutions are also applicable to the DNS.

2. Core Concepts

This section forms the foundation for the rest of the paper. In it, we explain the necessary components of the DNS which are required to understand its challenges and also the solutions discussed in the rest of the paper. Furthermore, it provides readers with deeper knowledge of the DNS protocol, necessary to discover other challenges and solutions that go beyond this article.

2.1. The Origins of the DNS

Standards for naming hosts on the network are almost as old as the Internet (or rather its precursor, the Advanced Research Projects Agency Network (ARPANET)) itself. Initially, every site connected to the early network maintained a copy of a file called `HOSTS.TXT` that provided a mapping from names to network addresses [27, 28]. The early pioneers realized that keeping separate copies of this file synchronized for a growing network was bad practice. This issue was finally addressed conceptually in the late 1983 by the first set of specifications for the DNS [29, 30] and transition plans to migrate from a centrally managed database of names to the DNS [31, 32, 33]. In 1987 the DNS specifications were updated, resulting in the basic protocol that is still in use today [34, 35].

Figure 2: Domain name concepts

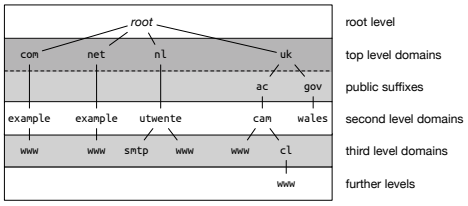


Figure 3: Example DNS hierarchy

2.2. Message format, components, and actors

This section provides a detailed discussion of the core concepts of the DNS. Throughout this section and the remainder of this article we will use the DNS terminology as specified by the IETF DNS operations working group [36]. Furthermore, definitions of aspects of the DNS protocol originate from the original DNS specifications [34, 35], unless specified otherwise.

2.2.1. The Structure of the DNS

Domain Name Concepts and Terms. The central concept in the DNS is the *domain name*. A domain name is represented as a structured ASCII character string. In this representation domain names are built up from labels separated by dots. Figure 2 shows examples that illustrate domain name concepts. The left-hand side of the figure introduces the following terms relating to domain names:

- **Label** – Domain names are composed of labels, where each label is limited to a maximum of 63 characters in length. Labels may contain the letters `A-Z`, `a-z`, the digits `0-9` and the hyphen (`-`). Labels are *case insensitive*, that is: `www` and `WWW` are equivalent. In DNS messages labels are encoded using a single unsigned byte value that indicates the length of the label, followed by 8-bit ASCII characters for the label text.

- **Root Label** – The root label terminates a domain name and is represented by an empty label. In a textual hostname, the presence of the root label is sometimes indicated by a single dot at the end of the name, but this dot is often omitted. In DNS messages the root label is represented as a single byte value set to `0x00`. This label indicates the top of the DNS hierarchy (which we discuss below). Parsers of DNS messages must stop processing a domain name when they encounter the root label.

- **Hostname** – This term sometimes refers to the left-most label of a domain name (in which case it typically refers to the local name of a system). In other cases, the term refers to the whole domain name. Because of this ambiguity, we try to avoid use of this term in this paper.

The right-hand side of the figure shows the following terms:

- **Fully Qualified Domain Name** – Sometimes abbreviated to FQDN, this term means the whole domain name, i.e., all labels that make up the name, including the root label. This term is often used interchangeably with the shorter "*domain name*". In this paper, when we use the term "domain name" we generally refer to an FQDN.

- **{cc | g}TLD** – The acronym TLD is short for *Top-Level Domain*. TLDs are the domain names directly below the root in the DNS hierarchy (as will be discussed below). The terms *ccTLD* and *gTLD* are also frequently used. In the former, "cc" refers to *Country Code*, as these TLDs are specific to geographic countries. In the latter, "g" refers to *Generic*. Generic TLDs are, as the term implies, not specific to a country, and include, for example, `.com`, `.net`, `.org`, etc.

- **Public Suffix** – As we will explain below, some TLDs divide the namespace under their control into separate branches. The combination of the branch label and the TLD label is often referred to as a public suffix. There is even a publicly available list of such suffixes[2].

The DNS Hierarchy. The DNS has a hierarchical organization, shaped like an inverted tree. Figure 3 illustrates this showing a part of the actual

[2] `https://publicsuffix.org/`

DNS tree. At the top of the tree is the root of the DNS. The root of the DNS is managed by the Internet Corporation for Assigned Names and Numbers (ICANN). They delegate responsibility for the maintenance of top-level domains, shown directly below the root, to so-called *registries*. Some registries divide the namespace under their control into separate branches (public suffixes), as is for instance shown in the figure for the `.uk` ccTLD, with, e.g., a `.co.uk` for commercial domains and, e.g., a `.ac.uk` for academic institutions.

The next level down in the tree are second-level domains. These are the domain names that generally belong to people or organizations. Below second-level domains we find third and further level domains. There is no formal convention for how this part of the namespace is organized, although there are common practices. As Figure 3 shows, for example, it is highly likely that there is a `www` label to indicate a World Wide Web service.

The Domain Name Industry. Initially, the number of top-level domains in the DNS was very limited. In 1985, the first ccTLD, `.us`, was added to the DNS, soon followed by further ccTLDs. The names of ccTLDs are based on ISO-specified country codes[3] [37]. Initially, domain name registrations were handled centrally through Internet Assigned Numbers Authority (IANA). When Internet growth really took off, in the 1990s, this no longer scaled. This led to the introduction of a tiered model, where TLDs have *registries*, that allow separate companies, called *registrars*, to sell domain names to interested parties. The owner or holder of a domain name is referred to as a *registrant*. For gTLDs, this model is mandatory; there is a common set of requirements for registrars of gTLDs, set out by ICANN, against which registrars have to be accredited [38]. For ccTLDs the registration policy is determined by the registry operator, and differs from ccTLD to ccTLD. We note that the registration and administration of domain names is often referred to as taking place through the *Registry-Registrar-Registrant* (or RRR for short) channel. This channel is separate from the DNS and uses its own protocols (e.g., the EPP protocol [39] for communication between registrars and registries). In the period between 2000 and 2012, ICANN introduced a limited number of additional gTLDs. In 2011, ICANN announced a new policy that effectively opened up applications for a potentially unlimited number of new gTLDs. Under this policy, well over 1000 new gTLDs have been added to the DNS since 2013. Included in these are many TLDs that contain non-ASCII characters, so called Internationalized Domain Names (IDNs). While ICANN has not yet launched a subsequent round to admit new gTLDs, there is a lot of pressure from stakeholders to admit further new gTLDs (see, e.g., [40]).

Today, domain names are a multi-billion US dollar industry. The largest domain name registrar in the world alone, GoDaddy, reported a revenue of USD$ 1.5B in 2020 from just its domain sales business[4]. There are very few verifiable sources of the total turnover in the industry, but to give an indication, business intelligence firms quote revenues of USD$ 7B in the US alone in 2020[5]. In addition to the actors in the RRR-channel, managed DNS providers have entered the market as well. They provide services to manage the DNS infrastructure of a domain name which, traditionally, has usually been provided by the registrars or run by the registrants themselves (see Section 2.2.3).

Reverse DNS and Other Numerical Names. Generally, the DNS is used to translate human readable names into machine readable information. The reverse, however, is also possible. A *reverse domain name* can be constructed by taking an IPv4 or IPv6 address and reversing its numerical representation. DNS queries for this name can then be used to, for instance, find the name associated with an IP address (see also Section 2.2.2 below). Example 1 shows example mappings between IPv4 and IPv6 addresses and their corresponding reverse DNS names. As the example shows, for IPv4 addresses the name is simply a reverse of the dot notation of the address. For IPv6, the reverse name consists of all 32 nibbles of the address; as the example shows, this can be quite cumbersome.

What?	Example value	Example DNS name
IPv4 address	`93.184.216.34`	`34.216.184.93.in-addr.arpa.`
IPv6 address	`2001:620:0:9::1103`	`3.0.1.1.[...]0.2.6.0.1.0.0.2.ip6.arpa.`*

*Truncated to save space.

Example 1: Numerical DNS Name Examples

[3] With a few exceptions: `.ac`, `.eu`, `.su` and `.uk`.

[4] Source: GoDaddy Annual Report 2020

[5] `https://www.ibisworld.com/industry/web-domain-name-sales.html`

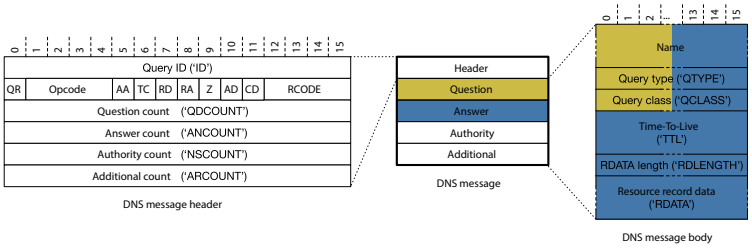


Figure 4: DNS message format, header layout and resource record format

2.2.2. Base DNS Protocol

Message Format. The DNS uses the same basic message format for all messages, with certain fields filled, depending on the message type. Figure 4 shows the DNS message format. The middle part of Figure 4 shows that a DNS message consists of a header, followed by four sections. The format of the header is shown in Table 2. Each of the four sections is filled with *resource records*. The general format of resource records is discussed in Table 3. In a DNS query, only the question and sometimes the additional section (see Section 6.2) contain information. In a DNS response, all four sections may contain information. The content of each section depends on many factors, including the response status of a DNS request (the `RCODE`). In general, each of the four sections has the following semantics (according to the original DNS specification [34]):

- **Question** – Contains the question in a DNS query (generally the name and type queried for).
- **Answer** – Contains the resource records that form the response to the question.
- **Authority** – Contains resource records pointing to authorities (name servers) for the queried name.
- **Additional** – Contains additional resource records pertaining to aspects of a DNS message, for example resource records with additional information – like glue records – on the authorities listed in the authority section.

In DNS responses, the answer, authority and additional section are all optional. Typically, though, in a successful response to a query, the answer section will contain one or more resource records that answer the query. In a successful response, the authority and additional section are usually optional,

Table 2: DNS Header Fields

Field	Description
Query ID	Identifies the query and helps match queries and responses.
QR	This flag indicates whether the message is a query (`0`) or a response (`1`)
Opcode	This field identifies the DNS operation. The most common value is `0` for a query/response operation (other values are assigned in [41]).
AA	This flag indicates whether the DNS response is an Authoritative Answer.
TC	This flag indicates whether the message was truncated because it exceeded the maximum message size.
RD	This flag indicates whether Recursion is Desired (explained in Section 2.2.3).
RA	This flag indicates whether Recursion is Available.
Z	Set to zero and reserved for future use.
AD	This flag indicates whether the response contains Authenticated Data (see Section 6.4).
CD	This flag indicates if DNSSEC Checking should be Disabled (see Section 6.4).
RCODE	The response status of the DNS request. Important values are `NOERROR` (`0`), `SERVFAIL` (`2`), `NXDOMAIN` (`3`) and `REFUSED` (`5`).
Question count	Indicates how many questions are in the question section. Currently, this field is always set to `1` in DNS queries and responses.
Answer count, Authority count, Additional count	Number of records in the Answer, Authority and Additional section.

Table 3: General DNS Resource Record Layout

Field	Description
Name	The domain name this resource record pertains to. Note that names may be compressed, to save space in datagrams. DNS compression works by replacing a label in a DNS name by a pointer to another DNS name in the same datagram. Compression is explained in more detail in [35].
Query type	The query type is an integer that indicates the specific kind of resource record. Common record types are discussed further on in the section.
Query class	An integer that indicates the query class. Historically, the DNS distinguished multiple classes of networks. These have, however, become obsolete over the years, and in almost all cases the query class is set to `1` to indicate the Internet (class `IN`).
Time-To-Live	The Time To Live (TTL) field is an integer that provides an indication how long (in seconds) a resource record may be cached. The use of this field is discussed in more detail in Section 2.2.3.
RDATA Length	This field indicates the total length of the resource record-specific data that follows.
RDATA	Variable length field with data that is specific to the resource record type.

that is: they may be left empty, for instance to save space in a DNS message.

Query/Response Protocol. DNS messages are normally transported using UDP, and the original DNS specification lists a maximum payload size for DNS messages of 512 bytes. The use of UDP means that most DNS exchanges are asynchronous and connectionless. In some cases, messages are exchanged over TCP. A typical DNS message exchange, in which a DNS client sends a query to a DNS server, and the server returns a response to the client, looks like this:

1. *Client sends query* – The client composes a query by filling the question section of a DNS message. In this section, the client indicates the name, query type and query class in which they are interested. The client sets the `QR` flag to `0` to indicate that the message is a query. The client optionally sets the `RD` flag to `1` to indicate that the client would like the receiving server to perform DNS recursion on its behalf (see Section 2.2.3).

2. *Server sends response* – The server responds to the request in the question section of the query. It copies the question section into the response, and fills the other sections of the response depending on whether or not it is able to answer the request. The server then sends the response back to the client.

3. *(optional) Fallback to TCP* – If the server cannot make a full response fit in a single DNS message, it will set the `TC` flag in the response over UDP. This is an indication for the client to retry the query over TCP to get the full response.

Typically, DNS clients will initiate a request to a DNS server over UDP, but there is no hard requirement to do so. They may also directly initiate a request over TCP. In addition to this, clients may keep the TCP connection to a server open and issue multiple requests in a single session. Generally, UDP is still the preferred way to transport DNS messages. The main reason for this is performance; setting up a TCP connection requires more network round trips, and keeping TCP connections open for long periods of time unnecessarily consumes resources on both the client and the server. There are changes to the DNS on the way, though. A workgroup focusing on DNS privacy has standardized DNS-over-TLS (DoT) [42]. In the standard, the authors suggest using TCP Fast Open [43] to reduce the overhead of using TCP. We further discuss DNS-over-TLS in Section 5.

Table 4: Common DNS Resource Record Types

Type	Description
`A`	Maps a domain name to an IPv4 address.
`AAAA`	Maps a domain name to an IPv6 address.
`CNAME`	Specifies an alias for a name. If a `CNAME` exists for a name, incoming queries for that name are translated into queries for the name that the `CNAME` points to. For example: if there exists a `CNAME` that maps the name `foo` to the name `bar`, then a query for the `A` record for `foo` will effectively be treated as an `A` query for `bar`. `CNAME`s may be chained, that is: a `CNAME` may point to another `CNAME`.
`MX`	Specifies Mail eXchange records for a name. These are the servers that handle incoming e-mail for a domain. Mail servers attempt to deliver e-mail sent to `user@example.com` to the servers specified in the `MX` records for `example.com`.
`NS`	Specifies the names of authoritative name servers for a domain name.
`PTR`	Pointer record from a domain name to another domain name. This record type is most commonly used for reverse DNS, to map e.g. IP addresses to domain names (see Section 2.2.1).
`SOA`	Start Of Authority record. The `SOA` record specifies metadata about a DNS zone, such as the serial number of the DNS zone. DNS zones are explained in more detail further along in the section.
`TXT`	Text record. `TXT` records may contain arbitrary text strings with a maximum length of 255 characters each. `TXT` records are, for example, used for the so-called Sender Policy Framework (SPF) [44], which is designed to combat e-mail forgery.

DNS Resource Record Types. Table 4 introduces the most commonly used DNS record types in alphabetical order. Only basic record types that are part of the original DNS specification [35] are listed, other record types, such as those used for DNS Security Extensions (DNSSEC), will be introduced in Section 6.

DNS Zones. Data for domains in the DNS is organized into so-called *zones*. DNS zones contain resource records under a certain name in the DNS hierarchy. Zones are represented using ASCII text in so-called *zone files*, the format of which is specified in the original DNS specification [35][6].

Example 2 shows part of a DNS zone file for `example.com`[7]. At the top, on line 1, the `$ORIGIN`

[6] The original DNS specification [35] refers to zone files as "master files".

[7] Line numbers are included for convenience, and are not present in an actual zone file.

```
1  $ORIGIN example.com.
   ...
   domain name   TTL (<>)   class   type    value
2  @             86400      IN      A       93.184.216.34          ← RRset #1
3  @             86400      IN      AAAA    2606:2800:...:1946     ← RRset #2
4  @             86400      IN      NS      a.iana-servers.net.    ← RRset #3
5  @             86400      IN      NS      b.iana-servers.net.    ← RRset #3
6  www           86400      IN      CNAME   example.com.           ← Alias
7  sub           3600       IN      NS      ns1.example.org.       ← Delegation
8  sub           3600       IN      NS      ns2.example.org.       ← Delegation
9  lorem.ipsum   86400      IN      A       127.0.0.1              ← Results in empty non-terminal
10 *.dolor         300      IN      TXT     "sit amet nec..."      ← Wildcard
```

Example 2: Example DNS zone snippet for `example.com`

statement tells whatever software parses the zone file that all domain names in the file are relative to `example.com`. In other words: the label `www` on line 6 should be interpreted as `www.example.com`.

Lines 2-5 show resource records for the so-called apex record of the zone. The apex record is signified using the `@`-sign in the zone file, and points to the origin of the zone (`example.com`). Lines 2-5 also show the concept of resource record sets, or *RRsets*. An RRset consists of all resource records of a certain class and type for a certain name (e.g., lines 4-5 are an RRset consisting of all `NS` records for `example.com`).

Line 6 shows how a `CNAME` can be used to create an alias, in this case from `www.example.com` to the apex records of `example.com`.

Lines 7-8 show a *delegation* of a subdomain called `sub.example.com` to be managed by the two authoritative name servers specified (see also Section 2.2.3 below). Any queries for names in that subdomain should be directed to these name servers. Also note that on lines 4-5 there are `NS` records for `example.com` itself. This is not a delegation, these are the authoritative information on what the name servers for `example.com` are. In general, the delegation in a parent zone and the `NS` records in a child zone should be the same, but in practice these frequently diverge [45, 46]. This is mostly due to human error; administration of delegations is usually a very different process from editing of a DNS zone file. Especially delegations in TLDs are generally updated through the RRR channel, which, as we mentioned in Section 2.2.1, is completely separate from the DNS.

Line 9 shows a resource record consisting of two labels. This record illustrates that a DNS zone can contain multiple label levels under a delegation point. In this case, the zone thus not only contains records in `example.com` but also in `ipsum.example.com`. Because the subdomain `ipsum` is not delegated to other name servers, and because there are no records in the example zone for `ipsum` itself, this has another effect: `ipsum.example.com` becomes a so-called *empty non-terminal*. This has consequences for DNSSEC denial-of-existence proofs (Section 6.3.3).

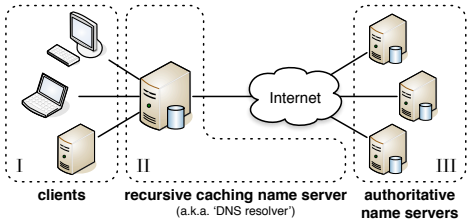


Figure 5: High-level architecture of the DNS

Line 10, finally, illustrates that the DNS also supports *wildcards*. A wildcard is always the left-most label in a domain name, and matches any label or labels provided (i.e., it also matches `<label1>.<label2>.dolor`). DNS servers will only return a wildcard record if the queried record does not explicitly exist. Thus, if, e.g., a record `nullam.dolor` is added, and a query is received for this name, that record will be returned rather than the wildcard.

2.2.3. Domain Name Servers

The DNS generally has two server roles. The first role is that of the authoritative name server, the second role performs DNS resolution.

Figure 5 shows these two roles from an architectural perspective. Authoritative name servers are

shown on the righthand side of the figure (III). The actors involved in DNS resolution are shown on the lefthand side of the figure (I+II). The two sections below explain these two roles in more detail.

Authoritative Name Servers. Authoritative name servers are, as their name implies, the authority for a domain. An authoritative name server can serve many domains. As mentioned in the previous section, authority for a domain is delegated to an authoritative name server in the parent zone of that domain, using `NS` records. As we will see in the next section, there is a delegation chain, from the root, which delegates to a TLD, which delegates to a second-level domain, and so on, and so forth.

If an authoritative name server responds to a query for a name for which it is authoritative, it indicates this in the response by setting the `AA` flag (Authoritative Answer, see Section 2.2.2). If a server is configured as only authoritative, and it receives a query for a name for which it is not authoritative and does not know of a delegation for the queried name to another authoritative name server, it will refuse the query by setting the RCODE in the response to `REFUSED`.

DNS Resolution. DNS resolution is the process of mapping a domain name to a value contained in the DNS. This process starts on the client (shown on the lefthand side of Figure 5). Say, for example, that a user wants to visit the URL `https://www.example.com/`. They type this URL into their web browser and press "Go". The first thing the browser will do is to attempt to resolve the address for `www.example.com`. To do this, it most likely calls a function of the operating system. Most OSes have a built-in *stub resolver*. This is a very limited DNS client that can send queries to DNS servers and can process responses returned by these. More importantly, however, is that a stub resolver cannot perform a process called *recursion*. That is: it cannot traverse the authoritative name servers in the DNS hierarchy to find a response to a query. Instead, a stub resolver typically sends a query to a *recursive caching name server* (shown in the middle of Figure 5). Recursive caching name servers are often referred to as a "DNS resolver", or simply a "resolver". Whenever one of these two terms is used in this paper, we are referring to a recursive caching name server.

As its name implies, a recursive caching name server performs a process called *DNS recursion* and

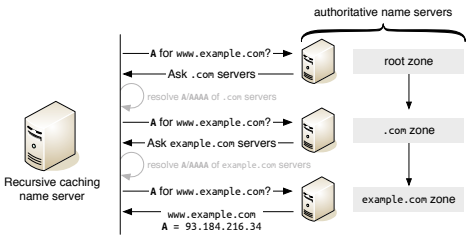


Figure 6: Example of a DNS recursion for `www.example.com`

it temporarily saves the results of this process in a so-called *cache*. Figure 6 shows the DNS recursion for `www.example.com` to continue the example from above. The figure shows the following steps of the recursion process[8]:

1. **Query to an authoritative name server for the root** – the resolver will start by sending the query for the `A` record for `www.example.com` to one of the authoritative name servers for the root of the DNS. The root name servers are operated on a vast, globally distributed infrastructure. The IPv4 and IPv6 addresses of the thirteen root name servers are well-known and preconfigured in most DNS resolver software. When a resolver first starts up it will typically perform what is known as a *root priming query* [47]. This means that it sends a query to one of the known root name server addresses to request the set of authoritative name servers for the root (`NS` query). It uses this to prime its cache with up-to-date information on the authoritative name servers for the root. Since the root name servers are not authoritative for `example.com`, they cannot respond to the query. The root, however, has a delegation for `.com`, and the queried root name server will respond with a list of authoritative name servers for the `.com` TLD, a so called *referral*. In other words: the root responds with *"I do not know, ask `.com`"*.

2. **Resolve addresses for `.com` authoritative servers** – a resolver with an empty cache will not have the addresses for any of the `.com` authoritative servers. In many cases, the answer from the root name server will include these

[8]Note that the figure shows a full recursion, which will only take place if none of the intermediate results required by the process are cached on the DNS resolver.

addresses in the *additional* section of the response it sends, and the resolver can use these. If the additional section is omitted, however, or incomplete, it will need to perform a separate recursion process to resolve these addresses in order to be able to query one of these servers.

3. **Query to a `.com` authoritative name server** – the resolver will now send a query for the `A` record for `www.example.com` to one of the `.com` authoritative name servers. Again, these name servers are not authoritative for `example.com` and will not know the answer. Thus, they will also respond with a referral, in this case to the delegation they have to one of the `example.com` authoritative name servers.

4. **Resolve addresses for `example.com` authoritative servers** – just like for `.com`, above, the resolver may not have the addresses for the `example.com` name servers in its cache, in which case it will perform a separate recursion process to resolve these addresses.

5. **Query to an `example.com` authoritative name server** – finally, the resolver sends an `A` record query for `www.example.com` to one of the `example.com` name servers. Since these are authoritative for the domain, they will return the requested response, the IPv4 address associated with `www.example.com`.

6. **Respond to client and cache** – once it knows the response to the query, the resolver returns the response it has learned to the client (in our example the stub resolver in the operating system) and it stores a copy of the response in its cache. This is also where the TTL comes into play, responses may not be cached for longer than the TTL specifies (but may, of course, be cached for a shorter period of time, for instance because a cache is full). Note that in case a resolver can answer a query from a client from its cache, it sets the TTL in the response to the remaining TTL, that is: the number of seconds that the record will remain in its cache. This ensures that cached records expire correctly if, for example, stub resolvers cache data, or when resolvers are chained such that one caching resolver forwards (most) queries to another upstream resolver. In the latter case, a resolver does not do the recursion itself but sends the query *forward* to another recursive resolver. This resolver will then look up the record and return the answer to the forwarder, which, in turn, will return the answer to the client. In principle, this chain of resolvers can be arbitrarily long.

Domain name	TTL	Class	Type	Value
`google.com.`	172800	IN	NS	`ns1.google.com.`
`google.com.`	172800	IN	NS	`ns2.google.com.`
`google.com.`	172800	IN	NS	`ns3.google.com.`
`google.com.`	172800	IN	NS	`ns4.google.com.`

Example 3: Delegation for `google.com` in the `.com` zone

In some cases, a referral needs to contain additional information to prevent situations in which a resolver is unable to continue the recursion. Take, for example, `google.com`. Example 3 shows the delegation for `google.com` in the `.com` zone. As the example shows, all four name servers are under `google.com` itself (this is sometimes referred to as *in bailiwick*). A resolver with an empty cache would be unable to resolve any name in `google.com` without knowing the addresses for any of these four name servers. However, to be able to resolve those addresses it would need to know the address for a name server for `google.com`, etc., etc. To remedy this circular dependency, the `.com` zone includes *glue records* with the `A` and `AAAA` records for the four Google name servers. If a `.com` authoritative name server returns a referral for `google.com`, it includes these glue records in the additional section of the DNS response.

These core concepts have helped the DNS to successfully grow in its over 30 years of history and they have not changed much. The deployment of DNS, however has been, and still is going through significant changes in recent years, which we explain in the next section.

3. DNS Evolution

In the previous section we explained the *concepts* of the DNS. In this section we explain how the different components are deployed *in practice* and what modern DNS deployments look like. Understanding how DNS is deployed in the real world is necessary to identify challenges and develop solutions.

3.1. Resolvers

In principle, each client can run its own resolver to query the DNS. Since the early days of DNS,

however, it was already recommended to run a central resolver within an organization [48]. This allows clients to save resources and to benefit from caching at shared recursive resolvers. This led to organizations and ISPs setting up their own recursive resolvers in their network, leaving the clients with stub resolvers, solely forwarding queries to upstream resolvers. This situation is also sketched in Figure 5. In the first decade of the 21st century, however, it became more common for users to choose a recursive resolver *outside* of their network. These *public* DNS resolvers promise additional features such as adult content filtering or increased performance. One of the early popular public DNS services, OpenDNS, reported that 1% of Internet users relied on its service in 2010 [49]. One year later, a study by Otto et al. [50] reported that 9% of Internet users relied on such a service. In the meantime, the complexity of recursive resolvers increased, now consisting of *pools* of resolvers to increase redundancy and performance [51]. We describe these more complex setups in detail in Section 7. These services turned out to be so reliable and trusted that users would turn to them in case their ISP's resolvers experienced problems [52]. Even ISPs themselves *forward* their queries to public DNS services today. By 2021, over 19% of Internet users rely directly on public DNS services [53, 54].

Even though the usage of public DNS services is not rising as fast as a decade ago, we expect that we will likely see even more users relying on *external* resolvers in the upcoming years. The reason for this is the rollout of encrypted DNS. Both browser vendors [55, 56] and operating system vendors [57, 58] are actively pushing for the encryption of DNS traffic. In the case of browsers, and in some cases this holds for OSes too, the stub resolver for encrypted DNS is implemented in an application, rather than at a central location in the core of the OS. These application-specific stub resolvers often connect to a third party DNS resolver by default. We discuss the technology behind encrypted DNS in more detail in Section 5 and discuss the pressures this puts on availability (Section 7) and the capability to detect malicious activities (Section 8).

3.2. Authoritative Name Servers

Right from the start, the DNS was designed such that zone content could be distributed among multiple authoritative name servers. A study from 2004 shows [45] that the majority of domains have two

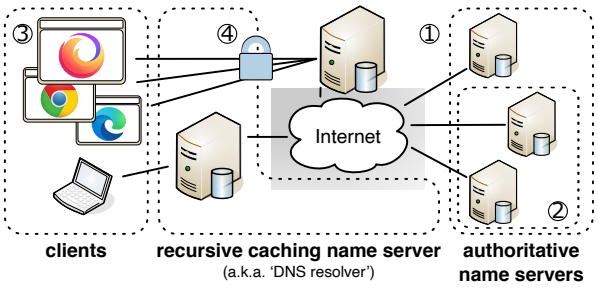


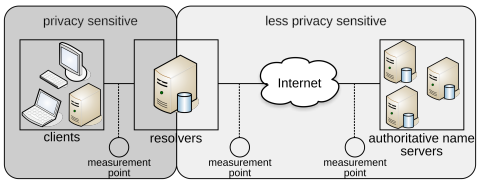
Figure 7: Deployment of DNS

or more name servers and we show that this is still the case today in Section 7.2.2. Traditionally, these name servers would be operated by the organizations owning the domain name, but this is increasingly less likely to be the case. A study by Shue et al. [59] shows that in 2007, some authoritative name servers are responsible for millions of domain names. Many of these domains were operated by DNS providers, and Hao et al. [60] showed in 2015 that especially social networks relied on these DNS service providers. This decision showed to be fatal, when one of the largest DNS providers Dyn got hit by Distributed Denial of Service (DDoS) attack in 2016, causing outages at many popular websites and services [6, 61]. As a consequence of this attack, some services chose multiple DNS providers to host their zone [62, 61], but despite this, the concentration of the DNS name space at a few providers has reached a new high in 2018, with a 25-fold increase over a period of nine years [63].

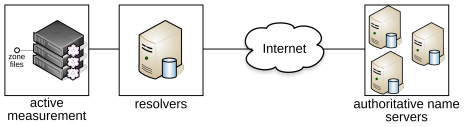
3.3. Modern DNS

Comparing the theoretical architecture sketched in Figure 5 with the real architecture anno 2021 as shown in Figure 7, highlights four important aspects of *modern* DNS. Instead of communicating with a recursive resolver located in the same network or in the network of the ISP, clients now often communicate directly with resolvers run by third-party public DNS service providers ①. Alternatively, the local resolver only forwards the queries to such a DNS provider. Also, ② instead of a distribution of name servers between different organization, many domain names are now under control of a few organizations.

In the future, we will likely see more and more software implementing their own stub resolver, bypassing the operating system ③. Clients will communicate encrypted ④, often with a recursive resolver of a DNS provider.



(a) Passive measurements



(b) Active measurements

Figure 8: Comparison of types of DNS measurements

Readers should keep these aspects in mind when reading the following sections. They cause challenges and explain why some solutions need to be implemented in certain ways. For example, encrypted DNS traffic, partially, provides confidentiality in the DNS, but on the other side hinders the detection of Internet abuse. We explain the details of these developments in more detail in the sections below, but first we explain, how we can measure the DNS in such an environment to understand the challenges and to develop solutions.

4. Measuring the DNS

Measuring the modern and real-life deployment of DNS is crucial in order to understand how the DNS is used, both by *bona fide* and by malicious actors. Measurements are also an important tool to identify challenges in the DNS and to understand how changes to the protocol and its use work in practice. In this section, we discuss the two methods for measuring the DNS: *passive* measurements and *active* measurements. Figure 8 shows a comparison between these two types of measurements and highlights the main difference: in passive measurements, DNS traffic is collected at one or more measurement points and observes traffic that is the result of DNS queries by real end users. In contrast, an active measurement precisely controls which queries are executed and collects results for these. In the remainder of this section, we explain the basics of both measurement types.

4.1. Passive measurements

Traditionally, DNS traffic is unencrypted, which enables passive observations directly at the client, resolver or authoritative name server and also on the path between those three. This approach observes the complete content of a query as well its response and gives a detailed view of DNS usage. Additionally, in case traffic between client and resolver, or between resolver and authoritative is encrypted, one can still observe DNS traffic on the resolver or authoritative host through the DNS server process, for example by using `dnstap` [64], which is supported by many open source DNS server implementations.

Figure 8a shows potential placement points for passive DNS sensors. Placing a passive measurement sensor directly at the client limits the observed traffic to queries from and responses to that particular client. In this configuration, we can capture *every* DNS packet which gives a complete view of the client's DNS traffic but only for that single client. An example is the study by Razaghpanah et al. [65] which monitor, among others, DNS lookups at mobile clients to study tracking ecosystems. In contrast, placing a sensor at a recursive resolver gives us insight into traffic of every client of this recursive resolver, which in case of a resolver at an ISP, can be millions of clients. We, however, also need to keep in mind that clients can configure multiple resolvers and a sensor at one recursive resolver might not gain a complete picture of a client's DNS traffic. Bildge et al. [66] analyze DNS traffic collected at ISP resolvers to detect malicious domain names. We can also place a sensor directly "behind" a recursive, on the path between the resolver and authoritative name servers on the Internet. This captures all DNS exchanges that are the result of so-called *cache misses*, and effectively collects the data for all names that clients of a resolver queried for, without identifying the specific clients that performed those queries. This protects the privacy of the users of the resolver, while still revealing what domains clients are actually interested in (we discuss privacy implications of passive DNS collection in more detail further down). This type of setup is, for example, commonly used by large-scale passive DNS collection services, such as DNSDB operated by Farsight Security[9]. Finally, we can also place a sensor at authoritative name servers, which will

[9] `https://dnsdb.info/`

```
root@localhost:~# tcpdump -vvv -i eth0 'port 53'
09:32:28.674038 IP (tos 0x0, ttl 64, id 63716, offset 0, flags [none], proto UDP (17), length 84)
        10.0.0.1.54584 > 8.8.8.8.53: [udp sum ok] 62669+ [1au] A? www.example.org. ar: .
        OPT UDPsize=4096 (56)
09:32:28.676558 IP (tos 0x0, ttl 60, id 27617, offset 0, flags [none], proto UDP (17), length 88)
        8.8.8.8.53 > 10.0.0.1.54584: [udp sum ok] 62669$ q: A? www.example.org. 1/0/1 www.example.org.
        [5h12m32s] A 93.184.216.34 ar: . OPT UDPsize=512 (60)
```

Example 4: Example of a passive measurement

give us visibility into every query directed from resolvers to that particular authoritative name server. Also here we need to take two pitfalls into account: first, a zone might be distributed across multiple authoritative name servers, so in order to receive every query directed to a zone we need to capture the traffic at every name server for that zone. Second, resolvers cache responses from name servers for some time and do not return to the name server until the cached response expires. This limits the number of queries seen by the name server. We explain caching in more detail in Section 7.5. Examples of studies relying on traffic collected at name servers include Castro et al. [67] analyzing traffic traces at the DNS root servers to gain a deeper understanding of the DNS ecosystem and Moura et al. [68] examining packet fragmentation, using traffic collected at a ccTLD.

A simple passive DNS measurement can be performed with the program `tcpdump` running on a client as shown in Example 4[10]. The executed command is shown on the first line. The second line shows the query from the client to the resolver asking to resolve the `A` record for `www.example.org`. Last, the third line shows the answer from the resolver. The client and resolver are highlighted in orange and yellow, respectively, and the query and answer are highlighted in blue and green respectively. The query of the client is repeated in the answer, as the third line shows. Tools exist to save and import the output of `tcpdump` directly into databases [69] or to extract parts of the information stored in DNS queries [70] for further analysis.

When performing passive DNS measurements, it is important to carefully consider the privacy of users. While domain names in and of themselves are public information, the specific query behavior of clients is very privacy sensitive. As such, the area

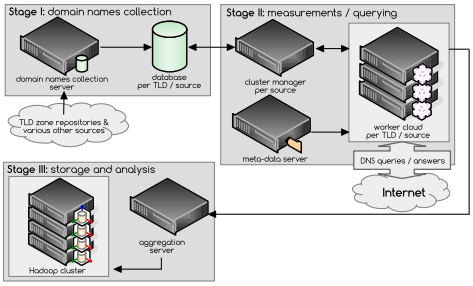


Figure 9: High-Level Architecture of OpenINTEL [10]

marked grey in Figure 8a is considered especially "privacy sensitive" since every query of a client is visible [71]. The difference between the left and right area is the information visible. On the left, every query from the client is visible but on the right, only queries which are not cached, by the resolver or by clients themselves, are visible [72]. This does not mean that traffic collected on one of the observation points shown on the right-hand side of Figure 8a is entirely devoid of privacy risks. Since the traffic observed "behind" the resolver is the result of cache misses, this traffic also includes queries for non-existent names, that may be the result of user errors (typing mistakes). The presence of queries that are the results of such mistakes may still reveal the presence of a certain user in the client population of the resolver behind which traffic is collected. IP addresses and the precise query string are collected in passive DNS measurements. Care should be taken when processing this kind of data and it should be anonymized if there are plans to make the data public. One trend in modern DNS deployments is the encryption of DNS traffic between client and resolver and resolver and name server. This improves privacy but also makes measuring DNS traffic on the wire almost impossible. We discuss encrypted DNS in more detail in Section 5.

[10] TCPdump displays the TTL as 5h12m32s rather than in seconds.

4.2. Active measurements

Besides passively measuring the DNS one can actively measure the DNS. In the case of active measurements, there are a number of considerations to take into account when designing and executing measurements. First, whether the measurement should create a one-time snapshot of the state of (parts of) the DNS, or whether the measurement should be longitudinal in nature. Second, whether the measurement should capture the resolved state of a domain, as seen by a client (this implies taking the output from a recursive resolver), or whether the measurement should capture the state of a domain on all of its configured authoritative name servers (this captures configuration errors and discrepancies). Finally, as with any active measurement, one should consider whether to perform a measurement from one or from multiple vantage points. In this section, our focus is on longitudinal active measurements from the perspective of a client. Nevertheless, we want to point readers to ZDNS[11] as a useful tool for snapshot DNS measurements. The benefit of having longitudinal data available is the possibility of uncovering trends in the DNS. For example, Toorn et al. [73] show the evolution of the use of TXT records over a period of three years. This is possible due to historic OpenINTEL data, which ranges back to 2015.

There are a number of active DNS measurements projects, such as 'OpenINTEL' [9, 10], the 'Active DNS Project' [74] and Netray[12]. These measure the DNS *in general*. Other studies focus on single aspects of the DNS, like `CAA` records [75], open resolvers [76], or DNS cookies [77]. In this section we use the OpenINTEL project to explain how active DNS measurements from the perspective of a DNS client can be performed since this is the longest running project, and the authors of this tutorial are involved with this project. A high-level overview of the architecture of OpenINTEL is shown in Figure 8b. In this paragraph we discuss the challenges of performing active DNS measurements at scale. We discuss two aspects, first performing the measurement itself and second part, the challenge of storing and analysing the resulting data.

The basis of an active DNS measurement is a list of domains which need to be measured (Stage I in Figure 9). Such a list typically comes from TLD zone files. In the case of OpenINTEL these zone files are acquired through TLD zone repositories and various other sources. With longitudinal measurements the frequency of the measurement is an important parameter, especially for larger sets of domains, since the measurement needs to be completed within the measurement period. Both OpenINTEL and the Active DNS Project have a measurement frequency of once per day. In order to finish the measurement before the end of the day, OpenINTEL measures the domains from the zone files in parallel via a swarm of workers – virtual machines tasked with querying domains (Stage II in Figure 9). The workers use off-the-shelf DNS software to perform the queries. This is important because it provides the best guarantees for the robustness of the measurement system.

The second challenge when performing active DNS measurements is the storing of the results. There are two consideration that need to be taken into account. First, measuring significant parts of the global DNS name space everyday generates sizable result sets. This means it is vital to choose a storage and analysis solution that is efficient. Second, if the measurement is expected to run for a long time (the OpenINTEL measurement is approaching the end of its fifth year), a storage format should be chosen that guarantees that future systems are also be able to read and use historic measurement results. This led the OpenINTEL project to choose Apache Avro as storage format, and the Hadoop ecosystem for analysis. A more detailed discussion of these choices can be found in the OpenINTEL design paper [10]. We note that, e.g., the Active DNS Project also chose to use the Hadoop ecosystem for storage and analysis [74].

4.3. Comparison

Passive and active measurements are both needed to understand and address the challenges of the DNS. Combining both measurements can give a more complete picture of the state of the DNS. Both types of measurements come with advantages and disadvantages, which we discuss in this section. You can use this information to decide which type of measurement best suits your needs for a particular study. We break the differences down into specific aspects of each measurement type, namely *privacy*, *confidentiality*, *coverage*, *frequency*, *complexity* and *availability*, in separate paragraphs below.

[11] `https://github.com/zmap/zdns`
[12] `https://netray.io/`

Privacy. As discussed above, privacy is an issue when performing passive measurements. The privacy impact depends on the observation point where traffic is collected; the closer to the client, the more privacy-sensitive the data collection generally is. For active measurements the privacy impact is very limited, as the DNS traffic is generated by the researcher performing the measurement. Somewhat related, though, is confidentiality, which we discuss next.

Confidentiality. This is a concern for both passive and active measurements. Due to the privacy sensitivity of passively collected DNS traffic, this type of data is not readily available in open repositories and typically requires researchers to enter into a contract with collectors of this data (e.g. DNSDB). While actively collected DNS data is typically not very privacy sensitive, it is often confidential. The main reason for this is that, e.g., operators of top-level domains often consider the contents of their DNS zones commercially sensitive, and hence require a contract for data access that limits to what extent this data can be shared. This is a challenge faced by active DNS collection projects such as OpenINTEL, the Active DNS Project and Netray.

Coverage. The coverage of passive and active measurements is typically one of the biggest differences between the two measurement types. Passive measurements observe DNS information resulting from a real interest by real clients, and thus better reflect real user activity. This is important, for example, in security-related research to detect emerging attacks and to estimate how widespread infections are. The biggest shortcoming of passive measurements is that they seldom cover the complete namespace of TLDs. Names that no client of the observation points where data is collected has shown an interest in will not show up in passive data sets. Consequently, for a better coverage, passive DNS measurement systems need many observation points, and it is likely that the law of diminishing returns applies to the extent to which this can grow coverage. In contrast, active measurements can cover entire name spaces, but only to the level to which the names are known. Thus, an active measurement, like the one conducted by the OpenINTEL or Netray project, covers entire TLDs such as `.com`, `.net` and `.org` for all second level domains in these TLDs. This means the actively collected datasets also include names in which users may not yet have shown an interest. Interestingly, the Active DNS Project also seeds their measurement with passively observed names, creating a mix of data.

Frequency. An aspect related to coverage is the frequency of observations. In passive DNS, there is no control over the frequency at which the same DNS query is observed, as this entirely depends on client behavior and caching behavior. Some queries will show up with very high frequencies – where there is a large client interest and/or a short TTL – whereas other queries will only sparsely show up. For active DNS, this is, of course, completely different, as the system that performs the measurement controls the query frequency. This makes active DNS more suitable for longitudinal studies where it is important that there are regular measurement results (e.g. one measurement every day), whereas passive DNS is more suitable for measurements where it is important to gauge the popularity of certain names and to identify, e.g., emerging threats.

Complexity. Both types of measurement have their own complexities. For passive DNS, it is challenging to find vantage points (due to privacy concerns), and to get good coverage requires access to a large number of vantage points. Furthermore, passive DNS data collection leads to high amounts of redundancy, as popular queries are typically observed at many vantage points. It requires careful consideration to cope with this. Active measurements, on the other hand, are difficult to scale up to significant parts of the name space, as illustrated by the papers about OpenINTEL [10] and the Active DNS project [74]. What both passive and active measurements at scale have in common is that even if redundancy is reduced, both generated significant amounts of data, requiring the use of so-called "big data" approaches to analyze the data.

Availability. As discussed in the previous two sections, there are a number of large-scale passive and active measurement projects. It is generally advisable, when starting a new study, to check the availability of the data collected by these projects, as in many cases they provide access to academic researchers. DNSDB is a very good source of passively collected data, and Farsight Security provides access to researchers under certain conditions[13].

[13] See the DNSDB website for more information `https://dnsdb.info/`

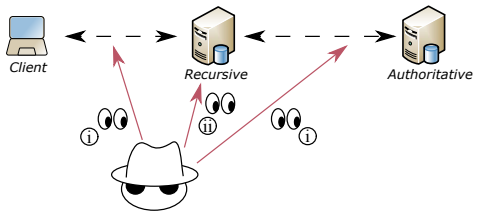


Figure 10: Challenges in the confidentiality of the DNS

More recently, researchers from Farsight in collaboration with academic researchers have established the DNS Observatory, to which researchers can also gain access [78]. For active measurements, as mentioned before, all three major projects, OpenINTEL, the Active DNS Project and Netray provide access to researchers under a contract.

5. Confidentiality

Having a solid background in the DNS (Part 1 in Table 1) we can now start looking at the challenges that modern DNS implementations face. We begin Part 2 by discussing the confidentially of the system. Figure 10 shows an overview of the challenges discussed in this section.

The original design goals of the DNS were redundancy, resilience and responsiveness - confidentiality was not considered at all. In fact, there was a long-held belief that "the data in the DNS is public" and therefore there was no need to consider privacy in the context of DNS.

In contrast, in terms of revealing the activities an end user is engaged in, DNS traffic is now recognized as one of the greatest privacy leaks on the modern Internet [4]. Because the DNS is an *enabler service*, DNS lookups precede almost all activities on the Internet, for example, visiting a website or using a specific service (e.g. chat, mail, applications). Historically all these queries have been sent in cleartext (i.e. using UDP or TCP on port 53). These queries form metadata about the end user (who can be identified by their IP address) and can be used to fingerprint, track or censor users very effectively, even if all the subsequent traffic is encrypted with e.g. HTTPS.

There are essentially two areas to consider in terms of DNS confidentiality. Firstly, the transmission of queries in cleartext on-the-wire, which can be subject to eavesdropping (i in Figure 10) . Secondly, the management of DNS data in servers (resolvers and authoritatives) and the sharing of such data by server operators with third parties (ii in Figure 10). Note that also the content of a zone file can contain sensitive information and must be protected [79]. In this section we focus on query confidentiality (we explain measures to protect the confidentiality of a zone in Section 7.2.1).

Recall that today clients generally use the OS's stub resolver to perform DNS queries. By default, the stub resolver sends queries to the recursive caching resolver that is provided automatically by the network via DHCP (the "system resolver") which is usually (but not always) within the local network and run by the network operator. We will refer to this type of resolver as a "local" resolver. Other DNS resolution services exist - "Google Public DNS" [80] is one well known global DNS resolution service operating on `8.8.8.8`. We will refer to these types of resolvers (those not associated with a particular network) as "remote" resolvers.

5.1. On-the-wire DNS Confidentiality

Whilst it is true (with caveats) that the data in the DNS is public, a single client DNS transaction *is not and should not* be public. A typical example from outside the DNS world is: the web site of Alcoholics Anonymous is public; the fact that you visit it should not be. The solutions available today to increase on-the-wire confidentiality involve both encrypting traffic between clients and resolvers, and minimizing the data sent from resolvers to authoritative servers. Note that this section only covers the most relevant DNS encryption protocols available. We limit ourselves to encryption protocols that have been standardised by the IETF and are thus likely to be implemented widely in application software and operating system network stacks. Non-standard solutions that are limited to few implementations, such as DNScrypt and DNScurve, are not discussed.

DNS-over-Transport Layer Security (TLS). The first protocol to be standardized by the IETF for encrypting client to resolver queries was DNS-over-TLS (DoT) in 2016 [42]. It uses port 853 instead of port 53 and it enables clients to set up long-lived TLS sessions to resolvers and then send multiple DNS queries over that encrypted connection (Opportunistic DoT). The connection can be also be authenticated if the client is additionally configured with a domain name for the service (Strict DoT).

Resolver operators were initially reluctant to offer DoT due to concerns about how it would perform at

scale, however three global anycast DNS providers offer DoT today (Google [80], Cloudflare [81] and Quad9 [82]). Several large ISPs are in the process of deploying DoT [83] e.g., Comcast in the US and British Telecom in the UK, and general deployment of DoT by other resolvers is slowly increasing.

Unfortunately the stub resolvers in the major desktop OSs (Windows and macOS) do not yet support user configuration of DoT and the only mobile platform to support it natively in the system configuration is Android. As a result the actual usage of DoT at the time of writing is mostly limited to Android users and privacy enthusiasts who choose to install desktop DNS privacy clients (e.g. Stubby [84]) or mobile apps (see Section 5.4). Notably, Apple allows applications to directly configure DoT and DNS-over-HTTPS (DoH) through their OS Application Programming Interfaces (APIs) since 2020 [85].

DNS-over-HTTPS. A more recent effort originated in the web browser community and resulted in a new specification for sending DNS-over-HTTPS (DoH) in 2018 [86]. This had several goals, but one was to enable applications to *directly* make DoH queries (rather than having to use the OS's stub resolver if it did not support encryption). This means that by using DoH individual applications now a have standardized option to do encrypted DNS to any resolver they choose, including remote ones.

A side-effect of using HTTPS on port 443 as the transport for DNS is that the DoH traffic is hard (in some cases impossible) to distinguish from other HTTPS traffic. Note that if for some reason a network operator wants to block DoT traffic out of their network to remote resolvers, they can simply block traffic on port 853. However, they do not always have the same option for DoH traffic.

This characteristic can be very useful for users in a "hostile" network where they may have genuine privacy concerns about their DNS. However, it poses a significant issue for many network operators who rely today on monitoring the network DNS traffic for many reasons including detecting attacks on the network, blocking malicious websites, scanning for malware or applying parental control filters.

The Firefox browser turned DoH on by default for all US users in September 2019 [55] (using Cloudflare as the default recursive resolver). This has sparked heated debate in the DNS operator community; many ISPs protested because such a decision had a sudden and dramatic impact on their operational practices. Discussions around Firefox's plans are currently ongoing with DoH recently being turned on for users in Canada in 2021 [87]. Section 5.2 discusses this and resolver choice in more detail.

Since then, Chrome followed suit in 2020 supporting DoH [88], although it employs a different policy whereby it automatically upgrades to DoH if the local resolver is on a managed list of recognized DoH providers. Similarly Edge also announce initial support for DoH in 2020. All these browsers also allow users to configure a DoH resolver of their own choice for encrypted DNS.

Also in 2020, Microsoft announced that DoH would be available as a configuration option in the Windows system settings. At the time of writing it is available in Windows preview only [58].

Besides DNS-over-TLS and DNS-over-HTTPS, DNS over Datagram Transport Layer Security (DTLS) has also been standardized as an experimental standard [89]. In contrast to the former, DTLS still relies on UDP instead of TCP, but is not widely deployed. Similarly, there is a draft specification for DNS transport over QUIC [90], but it is uncertain at the time of writing of this tutorial if that will progress to becoming a standard, as DNS-over-HTTPS will likely also support transport over QUIC with the introduction of HTTP/3 [91].

QNAME Minimisation. Look again at Figure 6 and notice that when the resolver sends the query to the root server it sends the *entire* query name, not just a query for the `.com` servers. It sends the same full query to every authoritative server in the recursion path. That means that every authoritative server in the recursion path (and any eavesdroppers on that part of the network) can see the full set of query names that the clients of each resolver send, unnecessarily leaking data.

To address this, a new recursive resolution method called "QNAME minimisation" [92] was specified in 2016 where the resolvers ask for *just* the part of the query name they actually need for that resolution step (so the root servers just get asked for `.com`, etc.). This does, however, actually introduce a few corner cases where name resolution becomes more complex. A simple example is to consider a case where a resolver needs to resolve `www.foo.bar.example` for the first time. Now, it is possible that there is a zone cut (an explicit delegation point) between `foo` and `bar` but not be-

tween `bar` and `example`, however a resolver cannot know this in advance. After obtaining the name servers for `.example` it will query those servers for the `NS` records for `bar.example`. However, it will get a `NODATA` response, indicating that there is no zone cut at that point, so it has to query the `.example` name servers again with one more label (`foo.bar.example`), and so on until it resolves the name if possible. Without QNAME minimization the required response would be returned immediately to the full query for `www.foo.bar.example`.

QNAME minimization can also increase problems querying authoritative servers that are not fully compliant with all the DNS specifications, especially regarding corner cases with empty non-terminals, like the example described above. At the same time, enabling QNAME minimization is becoming the default setting for most popular open source recursive resolver implementations and is increasingly deployed. An in-depth discussion of the current state of QNAME minimization and its pitfalls is given by De Vries et al. [93].

Keep in mind that these situations described above are corner cases. Essentially, QNAME minimization trades an occasional small latency increase in these corner cases for privacy, protecting the users behind the recursive resolver where DoT/DoH protects the privacy between client and resolver.

EDNS Client Subnet. One additional way that information can leak in queries to authoritative servers is if both the resolver and authoritative server use a technique called EDNS Client Subnet (ECS) [94]. Large Content Delivery Networks (CDNs) started using this technique to help them steer clients to the geographically closest source of content (to reduce latency); it works by adding a subnet of the address of the client that made the query into the DNS message sent from the resolver to the authoritative (for example, the first 24 bits of an IPv4 address). The authoritative server can use this information to map the client subnet to a geographic location and then select to return a geographically local IP address from an available set.

Whilst this may make the Internet faster for many users, it is clear that this is another privacy leak in the DNS, particularly if QNAME minimization is not used. Some resolvers (e.g. Cloudflare) do not currently use ECS in order to protect their clients privacy. Those that do use it should follow guidelines to minimize privacy and performance issue. For example they should use the smallest subnets that are operationally feasible, use efficient probing to detect support for ECS or maintain a whitelist of servers which support ECS. [94] also discusses in detail the impact on caching of using ECS e.g. lower cache hit rates and larger cache sizes. An example of the real-world privacy impact of ECS can be found in work by De Vries et al. [52].

5.2. Confidentiality of Data in DNS Servers

Even with the above measures in place the resolver operator still sees all the DNS queries sent to the server by each client. Resolvers may be operated by many different organizations in different network contexts: ISPs for use by their residential customers, companies for internal use by their employees (Enterprises), public locations for use by customers or visitors (coffee shops, hotels, public transport) and global corporations for public use (Google, Cloudflare, Quad9) to name but a few.

There can be two extremes: there are scenarios where the resolver operator may be highly trustworthy or have a direct contractual agreement with the user to provide certain services, in other scenarios the resolver operator may have reasons to want to collect (and even share or sell) data on users without obtaining any form of consent.

Some regions also have strict guidance around what can be collected (e.g. the General Data Protection Regulation (GDPR) in the EU [95]), however most do not. As of 2017, ISPs in the US are legally allowed to sell their customers' Internet browsing history without explicit permission [96].

As mentioned, there are legitimate reasons (monitoring, malware detection, etc.) for trustworthy resolvers to gather data, but how detailed the data is and how long it is retained for varies widely. Ideally, responsible operators everywhere collecting data for these reasons would adhere to principles similar to those laid out under GDPR. That is, they would employ "data minimization" (i.e. collecting only the minimum data required for the specific purpose) and also "anonymize" or "pseudonymize" the data to obscure client IP addresses (we refer to [97, 98] for such anonymization techniques).

Examples of some privacy policies including data retention specifics can be found at [99], [100] and [101]. For example, some large operators claim they do not log IP addresses at all and discard all data after 24 hours. A very recent Internet best practice document [102] provides details on a range of anonymization/pseudonymization techniques and their properties and also a template for

a document operators can use to publish their privacy policy.

Selection of System Resolver. All these concerns mean that in a privacy-conscious DNS world, the selection of the system resolver becomes very important. The "best" option may not always be the default, it most likely depends on what type of network the user is currently on (e.g. to trust the local network resolver or choose a remote one?), and which remote resolvers the individual trusts.

Sadly the published privacy policies of operators are often not particularly meaningful for the average non-technical user. Since average users do not normally understand the basics of DNS, let alone its privacy implications, it is a particular challenge moving forward to enable average users to keep their DNS query data safe on all networks.

Trusted Recursive Resolvers. With the increasing deployment of DoH it may become normal that every individual application on a client actually uses a different remote resolver. Firefox have used the term Trusted Recursive Resolver (TRR) to denote a list of resolvers that will be hardcoded into Firefox, because they meet a minimum set of criteria set out by Firefox [103]. One of these resolvers will be the default, and knowledgeable users can either choose a different TRR or manually configure their own chosen resolver.

Firefox argue that it is better for users to always use one TRR with a published privacy policy than a different resolver on each network. The reality is more complex; it may be better to use Google Public DNS than your local coffee shop's resolver, but when on an Enterprise or European ISP network the local resolver can typically offer more privacy and security than a third-party non-European based company. Additionally, the question of "informed consent" with respect to an application changing the default destination for its DNS queries is one that is not well understood and has not been tested legally to date. For example, if a household has parental controls enabled via their ISP what are the implications of a browser bypassing those controls?

This does mean that the future of client DNS resolution is likely to evolve rapidly. Potentially a user's DNS queries will be spread across many different recursive resolvers by different applications and the user will have no central point of control of their DNS resolution. The jury is still out on whether that will result in increased privacy or not.

5.3. Confidential DNS in practice

Today there are several options for encrypting DNS from end devices. As mentioned, Android 9.0 currently supports DoT natively and it will, by default, probe the network DHCP provided resolver on port 853 and use DoT without authentication if it is available. If you prefer to manually configure your Android device you can go to "Settings→Network & Internet Settings→Private DNS". You will see a dialog where you can switch from the default of "Automatic" to "Private DNS provider hostname". You then need to enter the name of your chosen provider e.g. `dns.quad9.net`.

To encrypt DNS from your desktop one of the most widely available clients is Stubby [84]. It can be installed via packaging on most operating systems and then manual configuration to override the default systems settings is required. This also depends on the operating system - details can be found in the online documentation on the project's website.

A list of large-scale DoT and DoH services can be found at [104].

5.4. Encrypted DNS and DNSSEC

Some people have argued that the use of authenticated, encrypted transports for DNS provides additional security as well as privacy. This is because the encrypted connection prevents on-path attackers from interfering with the DNS messages (one of the goals of DNSSEC). Browser vendors have presented this as a further reason to deploy DoH by default.

Whilst it is true that DoT and DoH do provide message integrity for the stub to resolver path, the end user is still entirely reliant upon the resolver operator fully and correctly implementing DNSSEC. If the operator does not, then the user is still at risk of receiving bogus DNS answers (it is interesting to note that Mozilla's TRR policy does not require DNSSEC). No browser today offers full client-side DNSSEC validation which, if available, would secure the entire DNSSEC data path from authoritative to stub regardless of the transport used. In reality the use of encrypted DNS does not negate the need for DNSSEC, they are two orthogonal and complementary solutions to improve the DNS.

5.5. Open Challenges

DoH/DoT resolver discovery. As noted earlier, current solutions rely on either pre-configured or user-provided credentials for known resolvers. So browser and system configurations for encrypted DNS still face a major challenge - how to securely obtain the credentials needed to authenticate a dynamically discovered resolver providing DoT or DoH? There is currently significant time and energy being expended on finding a general, adaptive encrypted DNS discovery mechanism, in fact, a new working group at the IETF was created in 2019 to specifically tackle this challenge [105]. At the time of writing the various use cases (e.g., home-/office/public wi-fi), upgrade models (e.g., auto-upgrade/user interaction) and security models (e.g., DHCP, PKIX) are all under active discussion.

Query confidentiality at resolvers. As noted earlier recursive resolvers will still see all the queries made by a user and even those that follow [102] retain the capability to monitor client activity or may be subject to e.g., warrants requesting user data. There are some more radical proposals for the stub to recursive stage of resolution that attempt to provide enhanced privacy for users. One proposal is "Oblivious DNS" [106] which replaces a single recursive resolvers with a pair of servers, each with a different role. The server closest to the client receives encrypted queries and forwards them anonymously to the second server. This second server can decrypt the queries and send an encrypted response back to the first server, which it then relays to the client. In this way, no one server has knowledge of both the client that asked the query and the content of the query. There is an obvious latency cost to this approach but without a change to the fundamental DNS model, the recursive resolver remains a clear control point, for better or worse. There is a specific variant of Oblivious DNS for DNS-over-HTTPS. This so-called "Oblivious DoH" (or ODoH for short) is under active standardisation in the IETF [107]. In late 2020, Cloudflare announced an experimental ODoH service for its `1.1.1.1` public DNS resolver [108].

Encryption to Authoritatives. We have focussed here on encryption between the stub and recursive resolver but even with QNAME minimisation there remains data leakage on the recursive to authoritative path which can compromise user privacy. The IETF DPRIVE WG [109] was re-chartered in 2019 to additionally focus on encrypting the recursive to authoritative stage of resolution. Again, discovery of authentication credentials is a significant challenge here - whilst a stub client is likely to use in the order of 10s of recursive resolvers even across several networks, a recursive resolver must be able to discover the credentials of *any* of hundreds of thousands of authoritative name servers it must contact in order to resolve the requests it receives if all upstream queries are to be protected. This is a non-trivial problem to solve.

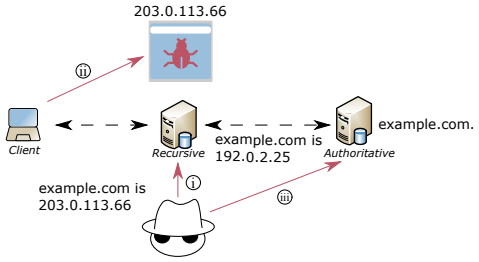


Figure 11: Challenges in the integrity of the DNS

6. Integrity

After attempting to solve the challenge of confidentiality, we are still not able to verify that a DNS response is authentic. In this section, we discuss how the authenticity and integrity of a DNS response can be verified. Figure 11 gives an overview of the challenges in establishing the authenticity and integrity of DNS responses and will be used as a guide throughout the first part of this section.

When the DNS was designed in the 1980s, security was not an important design consideration. At that time, the number of hosts on the Internet was still very small[14]. This meant that interpersonal trust between operators connected to the Internet made it unnecessary to protect the DNS against malicious activity. As the Internet grew, these implicit early assumptions ceased to hold. Given the importance of the DNS for the Internet, it was only a matter of time before the first attacks on the DNS showed up.

For example, in the Internet gold rush of the 1990s an organization called "AlterNIC" attempted to compete with the then legitimate administrators

[14]The number of hosts on the Internet crossed 1,000 in 1984, 10,000 in 1987 and 100,000 in 1989 [110].

of the DNS, InterNIC. AlterNIC hosted an alternative DNS hierarchy, in which they introduced new TLDs in which users could register domain names (at a lower price than InterNIC charged for domain names). This led to escalating discussions about who controlled what on the Internet, and eventually the founder of AlterNIC, Eugene Kashpureff, decided to hijack InterNIC's website through a DNS cache poisoning attack. This class of attack is particularly powerful, as its goal is to falsify information in the cache of a recursive caching name server (i in Figure 11), which in turn may have many clients depending on it. For example, residential Internet users typically use one or more DNS resolvers provided by their ISP. Injecting false information into the cache of such resolvers thus potentially affects all customers of the ISP. Cache poisoning attacks come in different forms and new variants still appear from time to time [5, 111, 112]. All have the goal to direct users to some unwanted website or service (ii in Figure 11), but their details are out of scope of this article. The main reasons such attacks succeed is that the DNS provides no means for resolvers to verify the authenticity and integrity of the data they receive.

Another common attack targets the content of a zone directly, e.g. by gaining access to the portal in which registrants configure the DNS-settings of their domain name using stolen credentials [113] (iii in Figure 11). Therefore it is necessary to not only protect DNS responses but also the origin of the response as well. In this section we will discuss how to do both using the DNS Security Extensions (DNSSEC). We first describe the history of DNSSEC and then explain how it works and how it prevents unwanted changes of DNS data.

6.1. A Brief History of DNSSEC

Development of DNSSEC started in 1994, when work started on the first RFC in an IETF working group tasked with developing the DNS Security Extensions. In DNSSEC, records in a DNS zone are digitally signed. If properly executed, digital signatures cannot be forged, and can only be created by the entity holding a secret cryptographic key. These signatures can then be validated by the recipient of a DNS message using the public key that accompanies the secret signing key. If the signature is valid, this proves the *authenticity* of the data in the message, as only the owner of the secret key could have created the signature. In addition to this, a valid signature also proves that the message

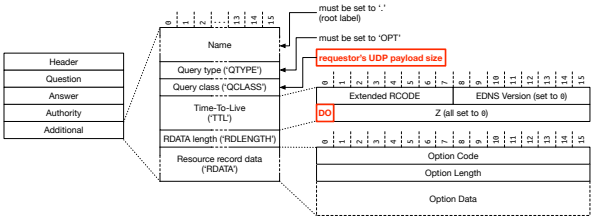


Figure 12: EDNS0 `OPT` resource record specification

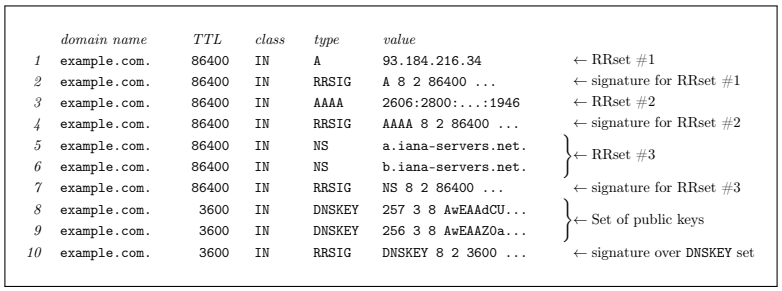
was not modified in transit (the *integrity* of a message), as any change to the content of the message would invalidate the signature.

The original specification did not see any significant uptake by DNS implementors and operators. There was, however, a lot of feedback on the specification from potential adopters, which eventually led to a set of three specifications [114, 115, 116] published in 2005 that still form the foundation of the DNSSEC protocol as it is in use today.

6.2. Prerequisites for DNSSEC: Larger DNS Messages

DNSSEC adds extra data to DNS messages in the form of digital signatures. As discussed in Section 2.2.2, the original DNS specification limits DNS message sizes to a maximum UDP payload of 512 bytes. This is problematic for DNSSEC. Take, for example, a DNSSEC message that includes one signature and a public key. If the key used is a 2048-bit RSA key, then the representation of the signature and key alone would already require more than 512 bytes. For this reason, DNSSEC relies on the so-called Extension Mechanisms for DNS, colloquially know as "EDNS0" [117].

The EDNS0 specification adds a way for DNS clients and servers to signal to each other that they have additional capabilities. This is done by including a pseudo resource record in the additional section of DNS queries and responses, the so-called `OPT` resource record. Figure 12 shows a detailed specification of the `OPT` record. Two fields in this record (marked in red) are important for the work in this paper. First of all, the requester's UDP payload size (encoded in the query class field). This value indicates to the recipient of an EDNS0 message what the maximum message size is that the sender is capable of processing. Second, the DNSSEC OK (`DO`) flag. If this flag is set in an EDNS0 message, it indicates the requester wishes to receive DNSSEC data if available.



	domain name	*TTL*	*class*	*type*	*value*	
1	example.com.	86400	IN	A	93.184.216.34	← RRset #1
2	example.com.	86400	IN	RRSIG	A 8 2 86400 ...	← signature for RRset #1
3	example.com.	86400	IN	AAAA	2606:2800:...:1946	← RRset #2
4	example.com.	86400	IN	RRSIG	AAAA 8 2 86400 ...	← signature for RRset #2
5	example.com.	86400	IN	NS	a.iana-servers.net.	← RRset #3
6	example.com.	86400	IN	NS	b.iana-servers.net.	
7	example.com.	86400	IN	RRSIG	NS 8 2 86400 ...	← signature for RRset #3
8	example.com.	3600	IN	DNSKEY	257 3 8 AwEAAdCU...	← Set of public keys
9	example.com.	3600	IN	DNSKEY	256 3 8 AwEAAZOa...	
10	example.com.	3600	IN	RRSIG	DNSKEY 8 2 3600 ...	← signature over DNSKEY set

Example 5: Example DNSSEC-signed zone snippet for `example.com`[7]

Just like DNSSEC, EDNS0 was defined to be backward compatible. Implementations that do not support the `OPT` resource record will simply ignore it, or return an error message. This should be interpreted by the sender of the EDNS0 message as a signal that the recipient does not support EDNS0.

6.3. DNSSEC Signing

DNSSEC signing is performed by the owner of a domain, and is done at the zone level.

6.3.1. Zone Signing

To sign a zone, the signer first needs to generate one or more cryptographic key pairs. The secret key of the key pair is used to create the actual signatures, the public key is included in the DNS zone. The public key will be used by recipients of DNSSEC-signed responses to validate the signatures included in the response. Keys are included in DNSSEC-signed zones using the `DNSKEY` record type. DNSSEC signatures always cover resource record sets (RRsets). The concept of RRsets was explained in Section 2.2.2 and is shown in Example 2. For each RRset a signature is included in the DNSSEC-signed zone.

Example 5 shows part of a signed zone for `example.com`. Lines 1, 3 and 5–6 show three RRsets, and lines 2, 4 and 7 show the signatures over each of these RRsets. Lines 8–9 show the `DNSKEY` records with the public keys with which the signatures can be validated. Line 10, finally, shows that all RRsets are signed, including the `DNSKEY` set.

Figure 13 shows a detailed view on the fields of the `RRSIG` resource record type. An actual signature record for `ietf.org` was used for illustration purposes. The following fields are shown, as specified in [115] (from left to right, top to bottom):

- **Type covered** – indicates the type of the RRset covered by this signature, in this case the RRset with `A` records for `ietf.org`.
- **Algorithm** – the signature algorithm used (in this case RSA PKCS#1 signatures [118] that use the SHA1 hash algorithm[15]).
- **Label count** – the number of DNS labels in the name covered by this signature. This field is used by validators to distinguish between signatures that cover a single name and those that cover a wildcard.
- **Original TTL** – the original TTL of the RRset covered by the signature. As discussed in step 6 of the DNS resolution process (Section 2.2.3), resolvers typically return the remaining TTL for cached records. Validators that use this data as input for validation must know the original TTL in order to be able to validate the signature [116].
- **Signature expiration** – the time at which the signature expires, and should be considered invalid. This time is represented in text as the concatenation of the year, month, day, hour, minute and second and on the wire as a UNIX epoch timestamp[16]. The signature in the example is thus valid until October 17, 2020 at 21:44:31h UTC.
- **Signature inception** – the time from which the signature is valid. The signature in the example is thus valid from October 18, 2019 at 20:44:52h UTC.

[15] An overview of current identifiers for DNSSEC can be found at `https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml`

[16] A 32-bit unsigned integer representing the number of seconds elapsed since January 1, 1970, 00:00:00h UTC.

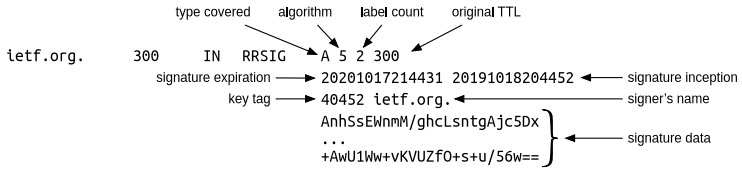


Figure 13: `RRSIG` fields

- **Key tag** – this 16-bit value is meant as a hint to validators which key should be used to validate the signature. Note that key tags are not guaranteed to be unique, so should only serve as a way to select candidate keys for validation.

- **Signer's name** – the owner name of the `DNSKEY` resource record that validators are supposed to use to validate signatures. Generally, this is the zone name of the zone the RRset covered by the signature is in.

- **Signature data** – the actual signature data, the format of which is specific to the indicated signing algorithm.

6.3.2. Chain of Trust

Before a validator can check the validity of signatures, it first needs to establish that it can trust the public keys it will use to perform the validation. It would be inconvenient if validators need to establish the trustworthiness of the public keys in every signed zone. Therefore, DNSSEC is designed to establish a *chain of trust* between parent and child zones. Normally, this chain of trust extends all the way from the root of the DNS down to an individually signed record. The chain of trust is established by creating a reference in the parent to a public key in the signed zone. This is done using the `DS` record type. The parent zone contains one or more `DS` records, that each reference a public key in the child zone that is a so-called *secure entry point*. This secure entry point is a key that is the root of trust for all signatures in a zone.

Figure 14 shows a detailed view of the fields of `DNSKEY` and `DS` records respectively. The example is based on real records for the `ietf.org` domain. For `DNSKEY` records the following fields are shown:

- **Flags** – 16-bit field that specifies properties of the key. The most important flags are encoded in bit 7 and bit 15. Bit 7 is always set on keys used to sign resource records. If bit 15 is set, this indicates that the key is considered a secure entry point. Note that it is not mandatory to set this bit, it is only intended as a hint to validators. In the example, bits 7 and 15 are set.

- **Protocol** – indicates the DNSSEC protocol used. This is always set to `3` for the current DNSSEC protocol [115].

- **Algorithm** – indicates the signing algorithm. Uses the same identifiers as the `RRSIG` record[15].

- **Public key data** – encodes the public key. The encoding is specific to the cryptographic algorithm that is used.

`DS` records have the following fields:

- **Key tag** – the key tag provides a hint about which `DNSKEY` record this `DS` refers to. The key tag is computed in the same way as the key tag in `RRSIG` records.

- **Algorithm** – the algorithm of the `DNSKEY` this `DS` record refers to[15].

- **Hash algorithm** – the hash algorithm used to create a hash of the public key data in the `DNSKEY` record this `DS` refers to[15].

- **Hash of public key data** – the actual hash of the public key of the `DNSKEY` this `DS` record refers to.

Figure 15 shows how this all fits together into a trust chain. The figure shows the trust chain for the `example.com` domain. Assuming that a validator wishes to validate the signature over a resource record for `www.example.com`, they can traverse the trust chain from that signature all the way up to the signing keys for the root of the DNS. What the figure also shows is the most common configuration for signed zones, which is to have two keys. The first key is called the *Key Signing Key (KSK)*. This key is the secure entry point for the

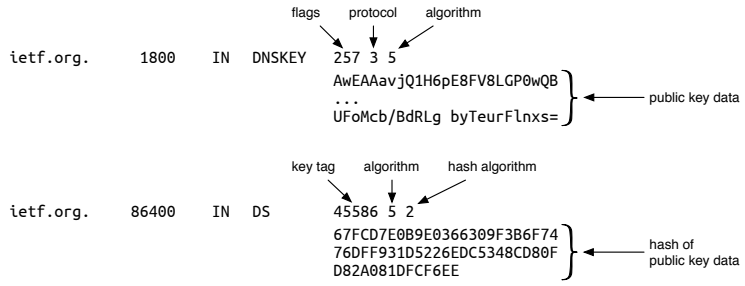


Figure 14: DNSKEY and DS record fields

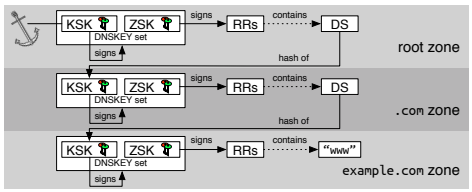


Figure 15: DNSSEC chain of trust for example.com

zone and is only used to generate a signature over the DNSKEY RRset. The second key is called the *Zone Signing Key (ZSK)*. This key is used to generate the actual signatures over the RRsets in the zone. The rationale behind this key management model is discussed below when we discuss the operation of signed zones. Note also that the trust chain always starts with a *trust anchor*. In almost all cases, the trust anchor is the KSK for the root zone of the DNS. Validating resolvers typically have the root trust anchor as one of their configuration parameters, as they need this anchor to be able to validate all other signatures.

6.3.3. Authenticated Denial of Existence

DNSSEC guarantees the authenticity of the records in a zone, and with that the presence of these records. But what if a record does not exist? In the DNS, name servers return a non-existent domain (NXDOMAIN) response if a record does not exist, but how can we trust that this is the case? DNSSEC also addresses this problem through a mechanism called *authenticated denial of existence*. Next to signing records in a zone, DNSSEC also generates cryptographically signed proofs of non-existence. These allow validators to verify that the name and record type in a query, for which they have received an NXDOMAIN response, do indeed not exist.

The original DNSSEC specification [115] defines the Next Secure (NSEC) record for this purpose. When signing a zone, the signer generates a chain of NSEC records that prove the existence of gaps in a zone, in which no names exist.

Example 6 shows a toy NSEC chain in the example.com domain[17]. The zone in the example is sorted in canonical order. For each of the RRsets in the zone, on lines 1, 3–4 and 6 the corresponding NSEC records are shown. On line 2, the NSEC record has two strings in its value field. The first string is the *next secure* name, in this case c.example.com. What the signer states with that is that no records for names between "a" and "c" exist in the example.com zone. The second string lists the types for which records exist that have "a" as owner, and in this case only an A record exists (shown on line 1). By showing what records exist, the signer implicitly asserts that no records of other types exist with owner name "a". The NSEC record on line 5 shows that no names exist between "c' and 'k" and that only an A and a AAAA record exist for "c". Finally the NSEC chain is completed by the record on line 7, which completes the circle by stating that no records exist between "k" and "a". Since "a" comes before "k" in canonical order, this wraps the chain.

Soon after the release of the DNSSEC specifications, there was criticism on the NSEC authenticated denial of existence approach. What makes NSEC problematic is that it allows so-called *zone walking*. By sending targeted queries for a signed zone that uses NSEC authenticated denial of existence, a querying party can establish which names exist in a zone, since the NSEC record conveniently lists the next secure (and thus existing) name. This means it is trivial to enumerate all records in a signed zone that uses NSEC. Some consider it undesirable that

[17] Note that for clarity, signatures are not shown.

	domain name	*TTL*	*class*	*type*	*value*	
1	a.example.com.	86400	IN	A	10.0.0.1	← RRset #1
2	a.example.com.	3600	IN	NSEC	c.example.com. A	
3	c.example.com.	86400	IN	A	10.0.0.2	← RRset #2
4	c.example.com.	86400	IN	AAAA	fe80::2	
5	c.example.com.	3600	IN	NSEC	k.example.com. A AAAA	
6	k.example.com.	86400	IN	TXT	"some remark"	← RRset #3
7	k.example.com.	3600	IN	NSEC	a.example.com. TXT	

Example 6: Example of NSEC records in example.com[7]

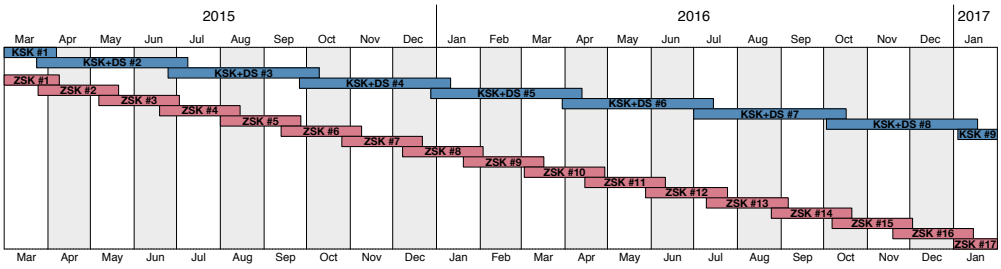


Figure 16: Key rollover for nobelprize.org (March 2015 – January 2017)

external parties can learn what names exist in a zone, as this may make it easier to perform targeted attacks.

To mitigate the problem of zone walking, an alternative mechanism was introduced, called NSEC3 [119]. In NSEC3, the next secure name is represented as a hash of an owner name, rather than the name itself. Canonical ordering of NSEC3 records is then done by ordering hash values. This means that trivial zone walking no longer uncovers names in the zone, but rather hashes of names in the zone. The details of NSEC3 and NSEC5, which addresses some shortcomings of NSEC3 are out of scope.

6.3.4. DNS Operation of a Signed Zone

As discussed earlier, DNSSEC signatures have a limited validity. This means that signatures need to be refreshed at regular intervals. In addition to this, best current practice suggests replacing signing keys at regular intervals [120, 121]. This has consequences for DNS operators that deploy DNSSEC signing. Where classic DNS only requires an occasional intervention to change a DNS zone, DNSSEC requires regular maintenance. Fortunately, there is mature DNSSEC software that can automate most of these processes.

First we look at re-signing DNSSEC zones. Most DNSSEC-capable software implementations can automate this process for operators. Typically, and as the best current practice [120] suggests, this means replacing signatures some time before they expire. This provides operators some respite if, for example, re-signing fails for some reason (e.g., a software crash).

Second, we look at the regular replacement of keys. This practice is called *key rollover* and the best current practice suggests that this should be done on a regular basis. Again, just like for signature refreshes, modern DNSSEC software supports automated key rollover. Figure 16 shows how this type of automation results in very regular key rollover patterns. The figure shows all key rollovers for the nobelprize.org domain between March 2015 and January 2017. KSKs are shown in blue, on top, and ZSKs are shown in red underneath. As the figure illustrates, the life cycles of keys overlap. New keys are introduced before they are used to create signatures, because it takes at least the TTL of the DNSKEY RRset before all caches will see a newly introduced key. If signatures with a key are published before the key has propagated to all caches, this can result in validation failures. Similarly, keys are removed some time after the last signatures generated with the key have disappeared from a zone. Again, this is to allow the TTL on RRsets with signatures made with the key to be removed to expire. A very important thing to note is that replacing a secure entry point into a zone generally requires interaction with the parent zone,

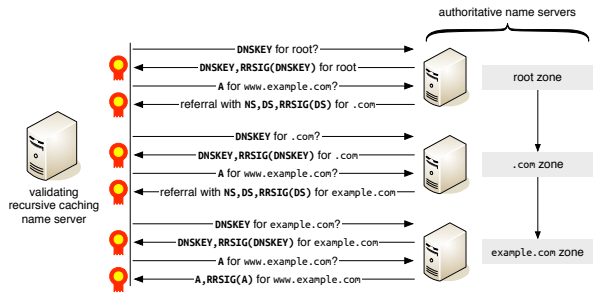


Figure 17: DNS recursion with validation for `www.example.com`

because the `DS` will also need to be replaced. If the parent zone is a TLD, then this interaction typically takes place through the Registry-Registrar-Registrant channel (see Section 2.2.1). This is one of the key reasons for having a split key model with a KSK and a ZSK. The KSK can be cryptographically stronger, and longer lived (requiring fewer interactions with the parent zone), and the ZSK can be cryptographically weaker, but replaced more often, independent of the parent zone. There are a number of possible variations for key rollovers, these are discussed in the IETF's best current practice [120]. Finally, we note that key rollovers introduce complexity into DNSSEC that threatens the resilience of the DNS. We discuss these issues and countermeasures in Section 7.

6.4. DNSSEC Validation

The final aspect of DNSSEC is the validation of signatures. This task is typically performed by DNS resolvers, which are then referred to as *validating resolvers*. A validating resolver needs a trust anchor to be able to validate signatures. As discussed in the previous section, this trust anchor is normally the KSK for the root zone of the DNS.

Figure 17 shows a DNS recursion for `www.example.com` together with the signature validations typically performed by a validating resolver. Each point where a validation takes place is marked by a red and gold ribbon. The pattern of validations at each level is similar, and conceptually consists of the following three steps:

1. **Fetch the `DNSKEY`** – the resolver needs to know the public keys to use to validate signatures in the zone.

2. **Check `DNSKEY` against `DS` and validate `RRSIG`** – to verify that the `DNSKEY` that was returned is the one in which the parent zone expressed trust, the resolver first checks if the `DNSKEY` set contains at least one key that matches a `DS` record in the parent zone. If that is the case, it validates the `RRSIG` record(s) in the response to the `DNSKEY` query to ensure that there is also a signature with a matching key over the `DNSKEY` RRset.

3. **Validate `RRSIG` in response to actual query** – finally, the resolver will validate the `RRSIG` record(s) in the response to the actual query using key(s) from the `DNSKEY` RRset.

There is one exception to this pattern. The root, of course, does not have a parent zone. Rather, whether the root zone returns the correct `DNSKEY` is checked against the trust anchor configured on the resolver. Validating resolvers will normally cache the result of signature validations, to conserve resources.

As discussed at the end of the previous section, DNSSEC current best practices suggest to replace signing keys regularly. For the root, this was scheduled to happen in October 2017 for the first time. Such a replacement has consequences for validating resolvers, as these need to switch to a new trust anchor. The IETF has specified a protocol for tracking trust anchors as these are replaced [122] and all modern implementations of validating resolvers support this protocol. However, when the scheduled date approached, doubts were raised whether the vast majority of resolvers have replaced their trust anchor [123]. These doubts were supported by a new protocol that enabled resolvers to signal to the operators of the root zone which key they have configured [124]. Results from the protocol indicated that many resolvers had not picked up the new key and analysis showed that, among others, applications using hard-coded trust-anchors were one of the contributors [125]. After a postponement of one year, for outreach and discussions within the community, the actual rollover took place in October 2018 and was considered a success. Nevertheless, a number of DNSSEC related outages were reported.

6.5. DNSSEC Deployment

Since the root was signed for the first time in 2010, DNSSEC adoption is rising, but is still far from being deployed widely. While, by the time of writing this article, 1,388 out of 1,527 TLDs (91%)

in the root zone are signed [126], reports claim that more than 90% of all 2nd level domains are not [127], with a few exception such as `.nl`, `.no` and `.se` where more than half of all second-level domains are signed.

On the validation side, measurements indicate that around 24% of users on the Internet are using a validating resolver [128]. This trend is also driven by the centralization of DNS infrastructure, as described in Section 3, where enabling validation at a few popular DNS providers is affecting many users at the same time. During the rollover of the root KSK, one thing that stood out was that more and more *applications* implement DNS resolution, and sometimes also validation [125]. This trend could make rollovers in the future more difficult since trust anchors are still not managed centrally on an operating system, which may pose challenges for future rollovers.

6.6. Zone files

While DNSSEC provides integrity for DNS response messages, the integrity of zone files (e.g., for a TLD such as `.nl` or `.com`) is equally important. For example, an attacker might change the `www` mapping for the domain `example.com` so visitors who log on to `www.example.com` unknowingly send their traffic through an intermediate server that the miscreant uses to record their usernames and passwords. Similarly, the attacker could also change the `example.com` mail settings so that the intermediate server receives and stores emails sent to `any_user@example.com`. Also, attackers could try to avoid detection by adding a subdomain under the existing legitimate domain (so called "domain shadowing" [129]).

Unauthorized zone file changes are often the result of a "domain registration hijack", which involves an attacker compromising an account of a registration provider (Registry, Registrar, or Reseller), for instance by using usernames and passwords obtained from other compromised sites. For high-value domain names, attackers also employ more advanced techniques such as spear phishing the staff of a registry to obtain account credentials [113]. Once the attacker manages to compromise the account, they use the registration provider's administrative panel to change the domain's records in the DNS, which eventually results in a changed zone file being conveyed to authoritative name servers.

For DNS providers, registration hijacks are an operational reality. For example, security company FireEye reported a global domain hijacking campaign in January 2019 that they claimed affected "*dozens of domains belonging to government, telecommunications and Internet infrastructure entities across the Middle East and North Africa, Europe and North America*" and where the adversaries for instance changed the IP addresses of domain names (the DNS `A` records) [130]. The Security and Stability Advisory Committe (SSAC) at ICANN provides an overview of compromises of registrant accounts that took place in the past and how they were carried out [113].

Registries, Registrars, and Resellers use various techniques to protect their registration systems against unauthorized changes, such as two-factor authentication and "domain locks", which are special services that require registrants to provide additional information to prove that they requested a domain name update. For example, the registry for `.nl` provides a "registry lock" that involves someone of the registry calling a registrar to confirm that the registrar requested to change the DNS records of a "locked" domain name.

Another way for an attacker to hijack a domain name is through "zone poisoning", which involves an attack on the zone file stored in a name server rather than on the registration system. Name servers that allow insecure dynamic updates of the zone file are vulnerable to this attack. If the attacker knows the name of the zone and of the name server they can replace records at will. By sending `UPDATE` requests to the name server, the attacker can, for example, update the IP address in an `A` record of a domain name, which has the same effect as a domain registration hijack. An initial measurement study by Korczyński et al. [131] carried out in 2015 – 2016 suggests that some 0.06% of domain names are still susceptible to zone poisoning. To prevent such an attack, operators should only allow updates from authorized sources, as described in Section 7.2.1.

6.7. Open Challenges

The measures discussed in this section are efficient when protecting the integrity of DNS messages and the zone file. However, there remain *open challenges* that require additional research.

Error prone deployment and maintenance. DNSSEC is on the rise at resolvers and domain names, but the majority are still not

protected. One reason for the lack of deployment is the concern of operators that DNSSEC is too difficult to deploy and maintain [132, 133]. For example, badly timed algorithm rollovers can cause validating resolvers to consider a domain name as *bogus* [120, 134]. Protocols to provide telemetry throughout a root rollover exist [124, 135], but their signals are hard to interpret and can be manipulated [125]. Also, validating resolvers are not able to differentiate between DNSSEC-related failures and other failures caused by misconfigurations, making it harder to debug these issues, even though a recent RFC could address this issue partially [136]. Thus, reducing the complexity of DNSSEC remains a major challenge and a roadblock for wider deployment.

Lack of client validation. Deploying DNSSEC validation at resolvers is only one piece of the puzzle. In order for stub resolvers to fully trust the DNS response they are receiving from their resolver, they would, ideally, implement DNSSEC validation themselves. Such stub resolvers exist [84], but are rarely deployed. One reason why DNSSEC validation at stub resolvers is missing is fear of additional round trip times and resolving failures due to misconfigurations and network middle boxes [137]. Addressing these bottlenecks could increase DNSSEC validation at clients.

Moving towards quantum-safe signing algorithms. Quantum-computers have the potential to break all signing algorithms currently used in DNSSEC in polynomial time [138, 139]. Even though it is not clear yet how long it will still take until this actually becomes feasible, it is important to evaluate *quantum-safe* algorithms for DNSSEC already. These algorithms can have significantly larger keys and signatures and may be problematic to deploy in DNSSEC. Also in general, transitioning to new algorithms takes time [140], thus it is necessary to start the assessment of post-quantum cryptography for DNSSEC early-on [141].

7. Availability

Confidentiality and integrity are only useful if resolvers are able to retrieve information from the DNS. Because the DNS is vital for the reachability of almost every service on the Internet, it needs to be especially resilient against outages. If a resolver or name server is unavailable, clients may not be able to reach some or all resources on the Internet. Furthermore, the current trend towards centralization of services on the Internet makes it even more important that these central components of the DNS stay available at all times. A crucial requirement, therefore, is a DNS which is reachable at any point in time. Attackers try to make DNS services unavailable, usually either by attacking the capacity of the individual servers (i in Figure 18) or the capacity of their connection to the Internet (ii in Figure 18).

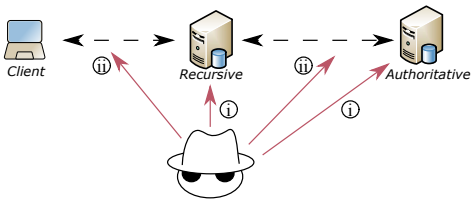


Figure 18: Challenges in the availability of the DNS

Table 5: Actions to increase DNS resilience

Action	@Authoritatives	@Resolvers
Sufficient Capacity	✓	✓
Multiple NSes	✓	✗
Anycast	✓	✓
Monitoring	✓	✓
Caching	✗	✓
Hardening	✓	✓

In this section, we explain how to deploy a DNS service that is as resilient as possible against partial or total outages and which actions operators of resolvers can take to be more independent from failures of authoritative name servers. Table 5 summarizes the measures operators can take that will be discussed in this section, and whether they apply to authoritative name servers, to resolvers or to both. The table lists these measures in the order in which they are discussed in the subsections that now follow. RFC 6168 [142] provides addtional best common practices not in scope of this article.

7.1. Sufficient Capacity

The first point of attention in the context of availability is that services must be provisioned with sufficient capacity. For the DNS, that means that authoritative name servers and recursive resolvers need to be able to receive, process and respond

to regular query loads but also to abnormal query loads, e.g., those caused by an attacker that wants to overload the server. The Internet community has long realized this need. Take, for example, RFC 2870 [143], from the year 2000, which recommends that DNS root servers should be able to handle at least three times their peak load. An implication of such recommendations is that both the hardware, on which the name server software is running, and the link that connects the server to the Internet need to be adequately dimensioned.

To put that in a modern context, recently, the developers of Knot DNS – an open source name server implementation – benchmarked the number of queries that a name server can respond to before dropping queries [144]. In this work, all major open source name server implementations were tested. The outcome was that every authoritative name server implementation can safely handle at least 500,000 queries per second, directed to a zone with 1 million domain names running on 5-year old hardware. The performance of recursive resolvers was slightly lower and developers of BIND report that their recursive resolver can handle between 400,000 and 500,000 queries per second and is heavily influenced by the number of clients and their query pattern [145]. A more recent run of the same benchmark on newer hardware showed the number of queries software can handle go up to 5 million queries per second without dropping traffic. Tools exist that operators can use to benchmark their own resolver setup based on real user traffic [146].

In addition to legitimate traffic, DNS servers, like other servers on the Internet, can be threatened by attacks that attempt to exhaust their link capacity through large volumes of unsolicited traffic. Considering that anyone can hire attacks that reach multiple gigabits per second for a few dollars [147], DNS operators should make sure that their links can cope with at least those traffic volumes. Recently, however, attacks have reached up to 1.3 Tbyte/s and it is not feasible to provide sufficient link capacity [148]. To stop such an attack from exhausting the resources of the server or the connection, operators can use special DDoS protection services that can filter extraordinarily high query loads and any other large volume Internet traffic [149].

While link capacity is important at the macro scale, link properties at the micro scale are also important for the DNS. In particular, limits on maximum packet size (e.g. imposed by firewalls) and Maximum Transmission Unit (MTU) size need to be taken into consideration. The size of a DNS message is limited to 512 byte in the original specifications [34, 35] but can be extended, using Extension Mechanisms for DNS (EDNS0) where 4,096 bytes is a typical limit. A study by Zhu et al. [150] finds that 75% of replies to queries for domains on the Alexa Top1000 are at least 738 bytes. Responses signed with DNSSEC increase the packet size even further. This needs to be taken into account when choosing the appropriate connection for the recursive resolver or authoritative name server. Concretely, it means that firewalls must not discard DNS messages over 512 bytes in size, and ideally they should also not discard IP fragments. If the latter is not an option, due to, e.g. fragment-based attack concerns, then resolvers and authoritative name servers must be configured appropriately to avoid fragmentation, by setting the maximum EDNS0 message size to below the MTU [151, 152], and, equally importantly, they must ensure that there DNS service is reachable over TCP. Recently, vendors of open source DNS software have jointly decided to change default EDNS0 parameters such that fragmentation should be avoided altogether [153].

DNS-over-TLS (DoT) and DNS-over-HTTPS (DoH) impose additional resource requirements on name servers and resolvers. Whereas traditionally most DNS queries are sent via UDP, DoT and DoH exclusively operate using TCP and encrypt and decrypt traffic. Zhu et al. [150] show that a resolver with 24k active TCP sessions requires 3.6GB of RAM and an authoritative name server with the same number of connections requires twice as much. Attackers may try to open more TCP connections to increase server load and operators should take countermeasures, e.g. limiting the total number of TCP connections and reusing existing connections [154].

7.2. Multiple Servers

Increasing the capacity of a single server and the use of DDoS protection services do not sufficiently reduce the chances of an outage. Luckily, the DNS allows the use of multiple recursive resolvers and to distribute a zone across multiple authoritative name servers. In that way multiple authoritative name servers share query load and stand in for each other in case of a partial outage. Authoritative name servers that are defined in a zone file are equal and a recursive resolver that requests information from that zone decides to which author-

itative name servers it sends its queries. Recursive resolvers use different strategies to select an authoritative name server but in general name servers that respond faster receive more queries [11]. When one of the authoritative name servers becomes unavailable, resolvers move on to another one [155].

Therefore, secondary name servers should be placed such that one network outage does not affect the availability of the zone [156].

7.2.1. Distribute the Zone

Every authoritative name server of a zone needs to serve the same content. If not, clients might be served outdated information, which, in case of a DNSSEC-signed zone, could even cause validation failures. Zones are distributed *in-band* using the DNS protocol and *out of band* using other Internet protocols or non-standardized APIs and user interfaces. Traditionally there exist two ways to distribute a zone to other name servers *in-band*: Authoritative Transfer (AXFR) and Incremental Transfer (IXFR) [157, 158]. With AXFRs it is necessary to transfer the whole zone, whereas IXFRs transfer only the changes since the last transfer. Both transfers are initiated by the server that wants to fetch the latest version of the zone, the so called *secondary*. It initiates a transfer by sending a query with query type `AXFR` or `IXFR` to the server from which it wants to fetch the zone, the so called *primary*. In case of an AXFR the queries are sent using TCP and the server will respond with every record in the zone, starting with the `SOA` record. When the last record is sent, the transfer is completed by sending the `SOA` record once more. In case of an IXFR, the requesting server can also use UDP as a transport and includes its latest known serial number from the `SOA` in its query. This helps the primary determine the difference between the old and current version of the zone. If the primary can fulfill the IXFR request, it will only respond with the difference between the version that the secondary has and the current state. Especially for large zones, IXFR can significantly reduce the time it takes to update the zone.

Usually, AXFRs and IXFRs are initiated by the requesting name server, e.g. after some timer has expired. With a DNS `NOTIFY` message the primary name server can also advise a secondary that the data of a zone has changed, and that a new version should be fetched [159]. `NOTIFY` announcements are DNS queries that include the `SOA` record of the updated zone, allowing a secondary to determine if it indeed needs to fetch an update.

Often, zone transfers should be limited to trusted name servers only. Therefore, primary and secondary name servers can share a secret key used for mutual authentication using the TSIG protocol [160]. TSIG relies on a resource record that is added to the AXFR and IXFR message, containing a cryptographic hash of the message content. Another common setup used to increase operational security is the use of a so-called "*hidden primary*". In this setup, the primary server is not visible to the public Internet and also not listed in the zone itself [120]. Because the primary name server is not reachable over the Internet, it is less exposed to external events that cause outages. Note, that a zone transfer is typically not encrypted and an attacker who is able to read traffic between the name servers could read the content of a zone during transfer. To address this, RFC 9103 [161] standardises transfers over Top Level Domain (TLD).

7.2.2. Picking the Right Number of NSes

To understand, how many name servers a zone should have we looked at common deployments in the wild. Using data from the DNS measurement platform OpenINTEL [162] we measure the number of servers that are configured at the root, TLDs, and second level domains of `.com`, `.net`, `.org` `.nl` and `.se`, as well as of domains from the Alexa Top 1M list [163]. The root servers have 13 different authoritative name servers. ICANN requires that each TLD delegated at the root has at least 2 different name servers, but most TLD operators choose to have at least 4 authoritative name servers. For second-level domain names, having two name servers is the most common.

The query load to each individual name server can be distributed with load-balancing software. Then, the IP address of a name server in a zone file points to a server on which a load balancer runs, not a name server. This software then decides, e.g. depending on the current load or query type, to which name server, hidden behind the load balancer, the query is forwarded. This can prevent individual name servers from becoming overloaded and increases the availability of a zone even further. In general, operators need to decide for each zone individually how much redundancy is required. As an absolute minimum, each zone should have at least different two authoritative name servers.

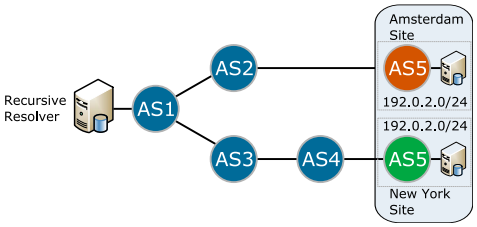


Figure 19: Example of a Name Server distributed across two anycast sites.

7.2.3. Redundant Resolvers

So far, we have mainly discussed having multiple authoritative name servers to host a zone. On the side of recursive caching name servers, however, redundancy is also important. Because of the impact of a recursive resolver failing – unavailability of DNS resolution services is typically perceived as "the Internet is broken" – it is generally highly advisable to make multiple recursive resolvers available to the users of a network. To this end, operators may want to set up two or more recursive resolvers in their networks and configure both at their clients. If one recursive resolver does not respond to queries the stub resolver that is part of the operating system of the client will automatically query the next configured resolver after some timeout. Also here, multiple recursive resolvers can be made available through a central load balancer.

7.3. Anycast

Besides deploying multiple name servers, each name server can also be replicated across multiple physical *sites* using *anycast* [164]. With anycast (and in comparison to unicast), multiple authoritative name servers share the same IP address, even though they are deployed at different locations around the world.

Figure 19 shows two name servers, both located in the same IP prefix but in two different physical locations. The Internet's inter-domain routing protocol, Border Gateway Protocol (BGP), is responsible for directing a DNS query from the recursive resolver on the left to one of the sites on the right. Because for this resolver the site in Amsterdam is reachable via only a single hop (AS2 in Figure 19) BGP's shortest path routing directs the query there. If the path to this site becomes unavailable, e.g. by congestion caused by a DDoS attack, the query is directed to the site in New York.

Such a setup can drastically increase the resilience of a DNS service against DDoS attacks and local network issues. For example, in 2015 the DNS root servers were targeted by a DDoS attack which caused some sites of the root to be overloaded. This caused traffic to shift and be distributed across other sites and kept service of the root servers online [165].

7.3.1. Placing the Nodes

Deploying and operating anycast name servers is more challenging than operating a few unicast name servers. First, an operator has to decide how many nodes an anycast network should consist of. The more nodes are spread across networks and sites, the more resilient the anycast network becomes. The placement of the nodes further depends on their optimal catchment. Ideally, each node serves an equal share of clients, such that load is distributed evenly. Global vantage points, such as the RIPE Atlas network, or tools like *Verfploeter* [166] allow operators to understand how many clients reach each node if they send a query to the name server. Operators can use techniques like AS-path pre-pending to steer traffic from clients from one catchment to another and thereby optimize their anycast network [167, 168].

In addition to providing sites that are reachable globally, operators can also configure anycast sites in a way such that they are only reachable from within certain networks. This practice is called local scope anycast [164]. By, for example, adding a special label (a so-called *community*) to the BGP announcement, operators can limit the propagation of their announcement. Such sites are isolated from the larger Internet, which limits their attack surface.

The increasing centralization of the Internet requires operators to place their nodes carefully. Moura et al. [165] observed during the DDoS attack against the root servers that even root server instances that were not attacked suffered from collateral damage because of shared infrastructure.

Today, all root servers and many authoritative servers of TLDs are distributed using anycast. The 13 root servers are distributed across between 2 and 194 sites [169]. Sommese et al. report that 97% of all TLDs and 62% of 2nd-level domain names in their data set use anycast at at least one of their name servers [170]. Some TLDs, such as `.nl` rely on a setup that uses a mix between global and local anycast [171].

7.3.2. Anycast at Resolvers

Next to authoritative name servers, recursive resolvers can also be deployed using anycast. Especially large public resolvers, such as the ones from Google or Cloudflare, use dozens of sites, all reachable under the same IP address to decrease response times for queries and distribute load [52, 172].

7.4. Monitoring

DNS service operators need to monitor the performance and reachability of their name servers continuously. This is especially necessary when a large number of servers and sites are deployed. Only then will they be able to detect attacks or configuration issues that can threaten the availability of their DNS service.

Under normal operations internal monitoring is necessary to monitor the utilization of the host machine as well as the performance of the name server software. The former can be done with tools that the host OS provides, for the latter tools such as DSC [173] and logs of the resolver software can be used. Operators should look out for everything "out of the ordinary", like an increase in query load (overall or for certain query types) or an unusually high CPU load. Additionally, we recommend monitoring DNS services *externally* as well. Measurement networks, such as RIPE Atlas, allow measuring response times and routing paths towards the name servers from thousands of vantage points, which gives a good indication how clients perceive the service from outside. During major operational changes, such as DNSSEC key rollovers, additional monitoring might be required. For example, Müller et al. developed a measurement methodology to monitor DNSSEC rollovers to detect failures early and thereby prevent that a zone might become unreachable [134].

7.5. Caching

Even in case no authoritative name server is reachable, information in the DNS might still be available because a resolver stores the response from a name server for some time in its local cache. In general, the more users share the use of a single cache, the bigger the fraction of queries that can be answered from the cache. It is not uncommon for the cache hit rate on busy resolvers to exceed 90%. The TTL of the received record defines how long a record may be stored in the cache, as explained in Table 3. Because the resolver is spared having to execute another recursion, not only the response time and the query load is reduced, but also the availability of the DNS is improved. As long as a resolver has the record cached, it does not matter whether the corresponding authoritative name server is available or not — the record can still be served.

7.5.1. Caching during a DDoS

The longer a TTL of the record, the longer an outage of an authoritative name server can last before the resource connected to the requested record becomes de facto unreachable. In the best case, if a resolver has fetched a resource record just before the authoritative name server becomes unreachable, clients can be served with the record for almost the whole period of the TTL. Resolvers are not forced to cache records for the whole time period of a TTL but can drop records earlier, e.g., because of limited capacity in the cache or other operational decisions. However, using a controlled experiment with more than 15,000 resolvers Moura et al. have demonstrated that 50% of observed resolvers implement caching in a way that allows clients to ride out outages that last no longer than the TTL [174].

7.5.2. Finding the right TTL

Operators that want to configure the TTLs of their DNS records need to find the right balance between a long TTL, which reduces query load and provides protection during a DDoS, and a short TTL that provides them with the flexibility to do efficient load balancing or to carry out operational changes in their DNS infrastructure. For example, operators might want to change the IP address of a web server in order to shift load to another one. This only works if resolvers query for the associated record frequently, which can only be achieved with low TTLs. Looking at TTL deployments in the wild, using our own measurements [10], we see that the operators of the root have configured a TTL of 1,000 hours for their NS records. This value is chosen because the authoritative name servers of the root zone almost never change and no dynamic changes to the zone are necessary. TLD operators use a median TTL of 24 hours, because they do not change the IP addresses of their name servers either. However the operators of the 1 million most popular domain names set the TTL of their `A` records to 300 seconds or lower in 50% of the cases. This low TTL allows them to shift load to different servers dynamically, but also makes them more vulnerable

in case their name servers become unreachable. After 300 seconds the `A` record is already dropped from the cache and clients would not be able to reach an underlying service (e.g. a website) anymore, even though the web server might be still available. Operators need to find the right TTL that is suitable for their use case and that balances agility with stability.

7.5.3. TTL at Resolvers

Recently, a standard has been adopted that allows resolvers to serve cached records beyond the TTL in case authoritative name servers become unreachable [175]. If a resolver that implements this standard receives a query for an expired record it first tries to fetch it again from the name servers. If that fails, it serves the *stale* record instead. Even before serving stale records became an official standard there are indications that operators already implemented it in their resolvers [174].

The effectiveness of caching may also be affected by the modern DNS extension "Client Subnet", used by content delivery networks to tailor responses based on the resolver's client [94]. With this extension, authoritative name servers can define for which networks a response is valid and resolvers respond from cache only if the query comes from a client located in the specified network.

7.6. Hardening

Finally, when the authoritative servers are deployed and the zone is distributed to every server it is important to harden them in a way in which they do not *enable* DDoS attacks as well.

7.6.1. Limit the Response Rate

DNS responses are usually larger than DNS queries which makes authoritative name servers attractive as an amplifier in a Denial of Service (DoS) attack. In such an attack, the attacker sends high amounts of queries towards a name server, using a spoofed source IP address of the victim. The response of the name server is then sent directly to the victim's IP, which can exhaust their capacity and take them off-line. We explain DDoS attacks in more detail in Section 8. To make name servers less attractive for being misused in such attacks, one common countermeasure is to limit the rate at which a server responds to queries, so-called Response Rate Limiting (RRL). In its most basic form, name servers keep track of the number of repeated queries per resolver. If one resolver sends too many repeated queries in a pre-defined time window (between a few seconds and several minutes), responses are dropped or truncated (forcing the resolver to fall back to TCP) [176]. This makes the name server a less attractive accomplice in amplification attacks.

7.6.2. Reduce the Record Size

Another countermeasure, that goes hand in hand with RRL, is to reduce the overall size of response packets. Because attackers want to achieve the biggest impact, they query for records that have the largest message size. Commonly abused query types are `ANY` queries. The response to such a query contains all records of every type for the queried name, including DNSSEC records. Because responses for DNSSEC-signed domains are typically the largest, attackers nowadays favor the use of DNSSEC-signed domains in amplification attacks. To reduce the response size of DNSSEC-signed records, operators can use modern, elliptic-curve-based algorithms that have smaller key-sizes and signatures [177, 178]. Moreover, some operators give severely reduced answers to queries of type `ANY` [179]. In Section 8.1 we describe in more detail how large RRs can be an enabler for powerful DDoS attacks.

7.6.3. Access Control

Especially for operators of recursive resolvers it is important to decide whether they want to serve queries from everyone on the Internet or just from a limited set of clients. Resolvers that serve queries for every client are called *open resolvers* and are often misused in the previously mentioned amplification attacks [180]. Unless an operator takes countermeasures, like RRL, we recommend limiting the clients that are allowed to send queries to a resolver. Also, accepting only queries via TCP prevents queries where the source IP address is spoofed. This is, for example, the case when resolvers support only DoH or DoT, since both protocols are TCP-only.

7.7. Open Challenges

All of the described approaches help to keep DNS highly available. However, 100% availability is very hard to achieve.

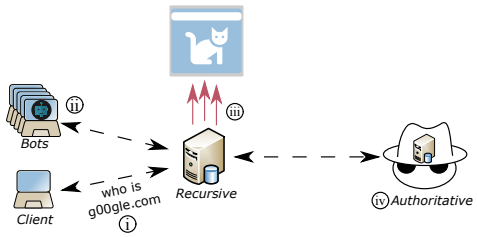


Figure 20: Challenges with the abuse of the DNS

Table 6: Role of DNS in attacks

	Attack facilitation	Communication	Exacerbation
DDoS	✓	✗	✓
Fraud	✓	✗	✓
Botnets	✗	✓	✓
Worms	✗	✓	✓
Spam	✗	✗	✓

Vulnerability to DoS attacks. The capacity and intensity of DDoS attacks is rising continuously. Planning enough capacity is thus not feasible. Techniques already exist to increase redundancy in the DNS across multiple servers or sites, but can hardly prevent local outages. Efforts exist to make the DNS itself more distributed [181, 175], but those are not widely deployed and are controversial.

Risks of centralization. Large DNS providers often have enough capacity to cope with larger DDoS attacks. However, if such a provider is failing, e.g. due to an attack, or configuration error, the impact can be significant causing outages of thousands of domain names and for millions of users [61]. Additionally, these providers have visibility into the query behavior of a large share of users, raising privacy concerns. In the last years, more and more domain names are hosted at a few service providers and a few resolving services become indispensable for the Internet [182]. Maintaining a distributed DNS is therefore crucial but remains an open challenge.

Realistic monitoring vantage points. Monitoring is crucial to discover outages or badly-performing instances. However, monitoring from the point of view of the clients remains challenging. Operators can, for example, rely on external vantage points [183], ICMP probes towards all IP addresses [166] in use, or TCP transmission metrics of the own client base [184]. These methods still give an incomplete picture, e.g. because of a limited number of vantage points or too infrequent measurements, therefore alternatives should still be explored.

8. Abuse

Before we look into how the DNS compares to other naming systems, we will discuss DNS abuse. The DNS is nowadays both the target of attacks (e.g. cache poisoning), and the means for conducting attacks (e.g. DDoS attacks). Security measures therefore fall in two categories: securing the DNS itself to increase confidentiality, message integrity and availability, or addressing the security implications DNS brings to the Internet as a whole. In this section we focus on the latter. We investigate the role DNS plays in attacks, and we highlight when the DNS provides information supporting the detection of these types of attacks. This follows the rationale that if an attack misuses the DNS in some way, these patterns are likely to become visible. We identify three possible roles the DNS can play in attacks (see Table 6): attack facilitation, in which the attack is impossible without DNS, e.g. misleading users to visit a phishing website (i in Figure 20); communication, in which DNS forms the communication medium, e.g. controlling a large number of bots (ii); and exacerbation, in which the DNS worsens the attack, by increasing the damage towards the victim, e.g. in a DDoS attack against a legitimate website (iii). In many of these cases, an authoritative name server is under the control of the attacker (iv). In the remainder of this section, we discuss these three categories to examine the role DNS plays in each of them.

8.1. Attack facilitation

The DNS plays a facilitating role when the attack becomes impossible to perform without the DNS. For example, some forms of DDoS attacks abuse DNS servers to reflect and amplify traffic to the victim.

In this subsection, we discuss two types of attack that are facilitated by the DNS, namely DDoS and fraud attacks.

DDoS attacks. DDoS attacks are a popular method of forcing a victim off-line, either by exploiting a protocol vulnerability or service functionality (semantic attacks), or by overwhelming the victim

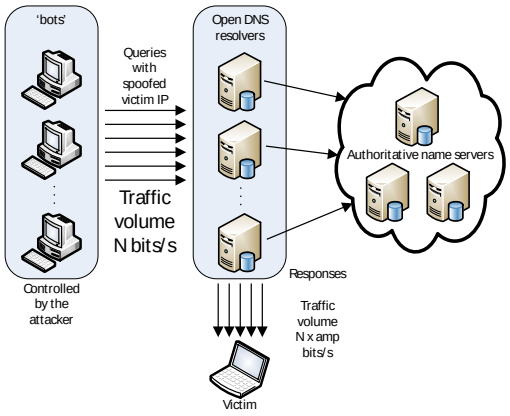


Figure 21: Example of a DNS reflection and amplification attack

with traffic (volumetric attacks). The DNS is misused in volumetric attacks. Figure 21 shows an example of such an attack, namely a *DNS-based reflection and amplification attack*. An attacker sends *spoofed* DNS queries to one or more open resolvers (reflectors) or authoritative name servers. The attacker often does so indirectly, using a swarm of compromised machines ("bots"). Running DNS over UDP allows the IP source to be spoofed in the DNS query. The open resolvers consequently send the response to the victim (*reflection*). It is important to note that the misused resolvers often engage in the full resolution process, if the response is not yet cached, which means that attack-related traffic trickles all the way up in the DNS hierarchy. A second factor in this type of attack is *amplification*. The amplification factor, defined in Equation (1), therefore plays a key role: a higher amplification factor means the attacker is able to induce more traffic for a given query size, in this way preserving their resources while leaving the burden of creating high volumes of attack traffic to the reflectors. The final volume of traffic sent towards the victim in a reflection and amplification attack is therefore influenced not only by the number of participating hosts (bots and reflectors), but also by the amplification factor. To summarize, the attack facilitating role of the DNS in a reflection and amplification attack is twofold. First, DNS resolvers act as reflector; and second, the data in the DNS enables amplification. We focus on these two aspects in the following paragraphs.

$$amplification\ factor = \frac{response\ size}{query\ size} \quad (1)$$

It is clear that open resolvers play a key role in DDoS reflection and amplification attacks. Without such resolvers, this type of attack would in fact not exist. This also means that open resolvers provide a way to measure DDoS amplification attacks in the wild. Researchers have approached this by creating fictitious open resolvers that minimally participate in the amplification but that are monitored to retrieve data on the attacks. Two projects are leading in this field. First, the University of Saarland *AmpPot* project [185] is a network of tens of amplifying honeypots [186] designed to track amplification attacks. The project focused on a plethora of protocols, among which the DNS, through passive measurement. Analysis of honeypot data allows to acquire a longitudinal view of amplification attacks, as well as information about, in the case of DNS, which queries and domains are misused. AmpPot data has supported a wide range of research, such as the attribution of DDoS attacks [186], the relation between reflection attacks and Booters [187] and in general the characterization of DDoS attacks [149, 188]. Similarly, the Cambridge Cybercrime Centre[18] runs a network of amplifying honeypots with the goal of monitoring attack behavior [189]. Both projects stem from the observation that attackers regularly scan the IPv4 Internet space for amplifiers (open resolvers in the case of DNS). The results discussed in Fachkha et al. [190] suggest that, beside scanning, high-rate DNS reflection and amplification attacks may also forgo the scanning phase, and only rely on a combination of spoofed requests to random IP addresses as reflector. With a sufficient number of requests, the attackers are in this way also able to trigger open resolvers.

An attacker can leverage the data in the DNS to achieve amplification by typically choosing a domain known to return large responses (e.g. a DNSSEC-signed domain, or a domain with a large `TXT` record size). DNSSEC itself (see Section 6) has been matter of debate if the benefits to integrity would not be accompanied by a higher potential for DDoS reflection and amplification attacks. Van Rijswijk-Deij et al. [177] investigated the effect of DNSSEC with regard to the amplification factor

[18] `https://www.cambridgecybercrime.uk/`

through analysis of active DNS measurements. This research compares the achievable amplification factor for DNSSEC-signed domains compared to unsigned domains, for a diverse set of query types. The authors define an acceptable upper limit for the amplification factor as the amplification factor achieved in regular DNS for the shortest query (`x.com` in the case of [177]) and the maximum response size (512 bytes in regular DNS). This leads to an acceptable upper limit for the amplification factor of approximately 22.3. The acceptable upper limit is used as a cut-off point between amplification inherent in the DNS itself and any other form of amplification due to DNS extensions. Query types individually fall within the acceptable limit except for the `ANY` query, both for unsigned domains as well as DNSSEC-signed domains. The average amplification factor of unsigned `ANY` queries is 5.9, while the amplification factor of DNSSEC signed `ANY` queries is 47.2. This leads to the conclusion that DNSSEC, although it can be misused, is not per se an enabler for DDoS reflection and amplification attacks. The paper moreover indicates how restricting `ANY` queries would already partially solve the problem of abusing (DNSSEC) domains. Due to the lack of legitimate uses, Cloudflare announced in 2015 [191] that it was phasing out the `ANY` query. Nowadays, no answer is received when querying `1.1.1.1` for type `ANY`. Google's open DNS resolvers, however, do respond to `ANY` queries, showing that opinions on the use of `ANY` queries are not unanimous.

A different way of achieving large responses is to query domains specifically crafted for this use. We have seen evidence, in active DNS measurements, of domains specifically crafted to ensure a large `ANY` size [192]. Figure 22 shows an example of this behavior and compares a crafted domain against a legitimate example (google.com). The figure shows that the number of records and the estimated amplification size (as carried, for example, in a query of type `ANY`) was modest until the middle of March 2015. From that time the domain has been inflated, specifically by adding more than 200 `A` records and reaching an estimated `ANY` size of 3,500 bytes. This coincides with the time window in which the domain is used in DDoS attacks, based on data from the AmpPot project [185]. After the attack window ends, in September 2015, the domain deflates. This behavior suggests that the rapid increase in the number of records within a domain is a sign of impending misuse in DDoS attacks.

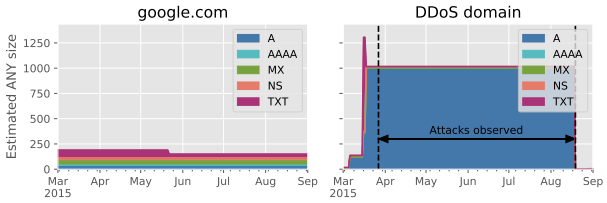


Figure 22: Evolution of a DDoS domain over time compared to google.com

Table 7: Examples of the different types of domain name squatting for the youtube[.]com domain name, from [193]

Domain Name	Squatting Type
youtube[.]com	Original Domain
youtubee[.]com	Typosquatting
youtubg[.]com	Bitsquatting
youtube-login[.]com	Combosquatting
yewtube[.]com	Homophone-Based Squatting
Y0UTUBE[.]com	Homograph-Based Squatting
xn--youube-k17b[.]com	IDN Homograph-Based Squatting (renders to youṭube[.]com)

Fraud attacks. The goal of fraud attacks is obtaining credentials of victims, or tricking them into transferring money towards the attacker. Typically, scammers clone an existing website, for example the website of a bank, to encourage victims to enter their credentials. The domain name forms a crucial part of these scams. Attackers aim to register a domain name which closely resembles the target domain. For example `g00gle.com` is visually similar to `google.com`.[19]

The practice of registering domains which closely resemble well-known domains is commonly called "domain squatting" or "cybersquatting" [194]. Common techniques within the field of domain squatting are listed in Table 7.

With typosquatting the attacker replaces characters from the target domain, relying on the notion of victims possibly making a typo when typing the target domain name. For example a user may type `ebau.com` instead of typing `ebay.com`. This type of attack relies on the fact that humans make errors, specifically when typing a domain name in, for example, their web browser.

The idea behind bitsquatting is similar to typosquatting. However, the mistake, in this case, results from a memory fault in the victims' machine. In principle domains created following the

[19] `g00gle.com` is owned by Google to prevent scammers from phishing attacks using this domain.

bitsquatting idea differ a single bit from the target domain: for example `youtubg.com` differs a single bit (compared to the ASCII character 'e') from the original `youtube.com`.

In combosquatting the original target domain is unmodified. Combosquatters either prepend or append words to the original domain. Combosquatting domains are notoriously difficult to detect because the practice of registering domains that closely resemble a target domain is not per se malicious, and companies use it either to diversify services, or to protect their own trademark from misuse. An example is the domain `youtubego.com`, that contains the trademark YouTube, but is not malicious [195].

The last class of squatting for Table 7 concern domains which are audibly (Homophone-Based) similar or visually (Homograph-Based) similar. The examples given for homophone and homograph based squatting are, for a human, discernible, with some effort. However, the introduction of Unicode characters in domain names (IDN) has made this squatting more complex to notice. IDNs allow Unicode characters to be used in ASCII-based domain names. There are Unicode characters which are visually indistinguishable from their ASCII counterpart. IDN Homograph-Based squatting, once rendered, is much more difficult to distinguish from the original domain, and the presence of a non-ASCII character can easily escape the user's attention.

The DNS facilitates cybersquatting attacks, because registering domains which closely resemble other domains is not prohibited. However, the practice is well understood and the ICANN itself supports registrants that become the victim of squatting attacks with a Uniform Domain-Name Dispute Resolution Policy (UDRP).

Detecting cybersquatting domains is typically done by analyzing the domain name itself. For example, Wang et al. [196] developed five models, reported in Table 8, with the aim of predicting which typosquatting domain may exist for a given target domain.

Bitsquatting domains may be detected by taking a target domain and evaluating all the permutations of bitflips for each character. Resulting domains which are not according to DNS specifications can be discarded. Following this approach, the authors of [197] were able to track the evolution of bitsquatting domains over the period of 270 days. During this period, they detected 5,366 different bitsquatting domains targeting 491 out of the Alexa Top 500 domains.

Combosquat domains are typically detected via a predefined list of trademarks. Because this type of squatting leaves the trademark intact, matching domain names with the trademark is an effective way of identifying suspicious domains. However, as mentioned, companies frequently register combosquat-like domains using their trademark themselves [193, 195], making distinguishing between malicious and benign domains difficult. Augmenting the domain names with additional data (e.g., from the DNS, but also Whois or Autonomous System (AS) information) may help distinguish legitimate from malicious domains. This is because legitimate domains are likely to be clearly linked to the AS and address space of the mother company, while suspicious ones are likely to be associated with other parts of the address space. Maroofi et al. [198] present a method for detecting defensive registrations which can be applied in this scenario.

By splitting words from a domain and replacing these with words from a homophone replacement database, the authors of [199] are able to detect homophone squatting domains with high accuracy. However, their approach was based on English dictionaries, making the approach ineffective for other languages.

Detection of IDN homoglyph domains is typically carried out based on the homoglyph character matching contained in so-called confusable tables (e.g. the *Unicode Confusables*[20] and *UC-SimList0.8*[21]). The work of Suzuki et al. [200] proposes an new confusable table, called *SimChar*, that extends the existing publicly available tables and that can be automatically extended if new homoglyphs are identified. The authors also provide a characterization of homoglyph registrations in the wild and how those are abused. Similarly, Yazdani et al. [201] propose a novel confusion table that builds on existing work and improves detection by a factor of almost 3× compared to state-of-the-art confusion tables.

The detection methodologies discussed here can make use of both passive and active DNS measurements, as long as the domain names are present. Active measurements may give a better overview of which squatting domains are registered, whereas passive measurements give an indication of the

[20] https://unicode.org/Public/security/

[21] http://people.csail.mit.edu/ayf/IRI/UCSimList/UCSimList/UC_SimList0.8.txt

Table 8: Generative models for typosquatting domains [196]

Name	Description	Example
Missing-dot typos	The "." following "www" is removed.	ww**wS**outhwest.com
Character-omission typos	Characters are omitted one at a time.	D**in**ey.com, Mart**hS**teward.com
Character-permutation typos	Consecutive characters are swapped one pair at a time.	NYTi**em**s.com
Character-replacement typos	Characters are replaced one at a time and the replacement is selected from the set of adjacent to the given character on the standard keyboard.	Di**d**neyWorld.com
Character-insertion typos	Characters are inserted one at a time and the inserted character is chosen from the set of characters adjacent to either of the given pair on the standard keyboard.	WashingtonPo**o**st.com, Googl**l**e.com

number of requests a squatting domain receives.

8.2. Communication

The DNS resolution process does not only map a name to the resource records related to it, but it can also be seen as a way of delivering information (communication) to the final user. Since DNS requests and answers are rarely dropped at a firewall, the DNS is an appealing communication medium for attackers. The DNS can therefore also play a "communication" role in an attack. In this subsection, we examine the communication role of DNS in botnets and worms (see Table 6).

Botnets. A botnet is a collection of hosts under the control of a botmaster. The botmaster typically controls the botnet through a Command and Control (C&C) server. The attacker can use the botnet to perform DDoS attacks, send spam, host phishing websites, or steal sensitive data. These modes of operation and common uses of a botnet are highlighted in the survey by Li et al. [202]. Computers may be infected by botnet malware through Web downloads, mail attachments, or through automatic scanning and exploitation of vulnerabilities. Understanding the method in which botnets obtain their commands from a C&C server is crucial in detecting botnets effectively. In the survey, three main communication models are identified. The first model is a centralized C&C model. Here a central point forwards messages between clients. The second model, is based on peer-to-peer communication. Messages are forwarded in a peer-to-peer fashion, between peers directly instead of via a central node, to each of the connected clients. This communication model is harder to detect than the centralized model, since no centralized host exists from where the communication originates. The third communication model, is an unstructured model. Here bots will not actively contact other bots or the botmaster, but the bots listen for incoming connections from the botmaster.

Getting commands across from botmaster to botnet is fundamental for the success of the botnet. There are two main requirements for this communication. First, the communication needs to happen covertly, as the botmaster aims at staying undetected for the longest time possible. Second, the communication needs be robust against takedown attempts of the C&C server.

Encoding C&C messages in DNS messages creates a covert channel for botnet communications. At the time of writing this paper, this practice seems to be rare, with the notable example of the DNSMessenger malware [203, 204]. Connections to the C&C server in the case of the DNSMessenger are always initiated by the infected host, making this a case of centralized botnet. However, the peculiarity of the DNSMessenger malware is that it leverages the DNS to set up a bidirectional covert channel with the C&C server. The malware encodes C&C connection and instruction messages in `TXT` records. According to the security analysts in [203, 204], this malware sample shows that botnet developers are willing to go the extra mile to remain undetected. DNSMessenger has shown that nowadays DNS traffic within corporate networks should also be considered among the standard protocols that might hide a covert channel, and therefore more attention should be placed on DNS monitoring.

Botnets that use the DNS as a communication method may be detected by analyzing the (DNS) behavior of the bots themselves, or analyzing the type of replies they receive. Both the behavior and replies can be observed in passive DNS measurements.

Arguably bots running the same malware will show comparable DNS behavior [205]. Such behavior may stand out in aggregated DNS data. For example, domains queried by multiple clients

at approximately the same time may indicate botnet behavior, especially if the query is for a non-existent domain. Furthermore, in the case where the C&C hides behind a Domain Generation Algorithm (DGA) domain (generated domains often based on, among other inputs, time, see [206] for an in-depth overview of different algorithms), bot behavior may be visible since a bot will send out many queries before contacting an IP address. An example of such a domain, generated by the Necurs DGA family, is 'bpwencsdvrjxji[.]pro' [207]. With DGAs the domain name may also be the basis of detection. Hoang et al. [208], for example, trained machine learning classifiers on features originating from n-grams of the domain name.

Another approach is analyzing the type of responses clients receive. Responses indicating name errors (`NXDOMAIN`, for example) may indicate a bot attempting to reach an older C&C domain, or a C&C server via a DGA approach [209]. Antonakakis et al. clustered hosts where their DNS queries exhibited abnormally large numbers of NXDOMAIN replies from passive DNS measurements.

These detection approaches typically use passive DNS measurements as they focus on the behavior of bots which is observable in passive measurements and not in active measurements. Detection of these algorithmically generated domains can be done via the analysis of NXDOMAIN responses [209], but also through analysis of individual domain names [210].

Worms. Worms are self-replicating pieces of malware code. A worm spreads over the network by exploiting vulnerabilities in target hosts. When a worm successfully infects a host on the network, the host will, in turn, try to infect other hosts. Worms typically contain malicious payloads; for example, the WannaCry [211] worm encrypts the hard drive after infection.

Spreading to other machines, in most cases quickly, is fundamental for a worm. In the IPv4 address space, it is feasible to randomly generate another IP address, or increment the current address, and use that as the next target. However, with IPv6 this is no longer feasible because the address space is too large. This is when the DNS comes into play. To circumvent this problem, a DNS worm generates plausible hostnames, similar to certain types of DGAs, and queries those. A successful query delivers the next target [212]. The DNS has therefore a supporting role in the worm's spread, and the system is used to communicate the necessary information (next target) to the infected host. Such activity may stand out due to stark difference in volume of DNS queries when compared to regular hosts.

The DNS usage patterns created while spreading form the basis for the detection of DNS-based worms. For example, Kammas et al. [213] theorized that spreading of worms via DNS queries will likely lead to NXDOMAIN replies, since the domain names queried for are generated and the chance is high that these domains are not registered. NXDOMAIN analysis is therefore an effective method to detect both botnets and worms using the DNS in the communication role. It is worth noting that also the lack of DNS traffic in combination with direct connections to other IP addresses can be considered indicative of malicious worm activity. Whyte et al. [214] observe that a connection to a host is typically preceded by a DNS request. They therefore propose to detect worm propagation by matching DNS traffic with outgoing traffic from a possibly compromised host. An IPv4 scanning worm, which does not need to rely on the DNS for spreading, would then stand out for the lack of DNS activity.

8.3. Attack exacerbation

Next to the "attack enabling" and "communication" roles, the DNS can play an "attack exacerbation" role. In this role the DNS improves the attack, typically increasing the damage of the attack. We classify an attack as using the DNS in an attack-exacerbating role when the attack can be performed without the DNS, but profits in some manner from the presence of the DNS. For example, the DNS provides a stealthier mode of operation, increases the damage, or makes the attack appear (more) legitimate. In the following, we highlight how the DNS can exacerbate the attacks previously discussed in this section – DDoS, fraud, botnets, and worms – and additionally consider the role of DNS in spam.

DDoS. The DNS exacerbates DDoS attacks in two ways. First, being a connectionless protocol, the DNS is susceptible to spoofing, which makes a DDoS attack harder to trace. Secondly, the data in the DNS may lead to large responses, which can be misused in amplification attacks.

Fraud. The DNS makes a fraud attack more believable. Receiving an email linking towards an IP address is suspicious. Receiving an email linking

towards a domain name is less suspicious, and may appear legitimate. In addition, the use of a domain name adds flexibility for attackers, who can, for example, easily migrate the malicious host to another IP address without invalidating the fraud campaign.

Botnets. Also in the case of botnets, the DNS exacerbates the attack by allowing C&C operators more flexibility. In the past, blocking access to the IP address of the C&C server was common. With a domain name in the bot's code this approach is ineffective, since the blockage can be subverted by a change in DNS records. Additionally, when generated domain names are used to resolve the IP address of the C&C server, it is extremely difficult to block access to the C&C server preemptively. A notable effort in this direction is the DGArchive [206], a web service that offers forward generation of malicious domains for several DGA families.

Fast-flux domains – domains where all the records exhibit extremely short TTL values, often less than five minutes – are usually strongly related to botnet activity and fraud, and play a role in hiding C&C servers. These domains can react quickly in the event of blocking of the C&C IP address, as bots quickly learn the new address due to short record lifetimes. Fast-flux domains are often employed by cybercriminals performing fraud attacks to protect, organize and sustain their scam service infrastructure [215]. Fast-flux can therefore be seen as a way the DNS exacerbates botnet and fraud attacks by supplying a highly dynamic platform of communication. A fast-flux domain is distinguished by the low TTL value of its records. However, this characteristic is shared with Content Delivery Network (CDN) services which typically also utilize low TTL values. The main difference between CDN- and fast-flux domains is where the records point to. A CDN service usually owns their own address space, and points their records to this space. Fast-flux domains, on the other hand, often point to hijacked hosts, meaning that the records do not point to the same address space, or address space owned by the same company. Clustering domains with low TTL values together and analyzing the destination of the records may detect fast-flux domains [216]. Table 9 gives an overview of the main differences between a "standard" DNS domain, a CDN domain, and a fast-flux domain. Yadav et al. [217] presented, based on the analysis of several

Table 9: Fast-flux domain characteristics [218]

	Standard DNS	CDN	Fast-Flux
A Records	4	4	4
NS Records	2	2	2
Network Ranges	1	1	3
Unique ASNs	1	1	2
TTL	$> 1{,}800$	$< 1{,}800$	≤ 600

(active) datasets, a methodology which could be used to detect domain fluxing activity within ones network. The alerts of the system can be used by a network security analysis to perform additional forensics to infer the exact algorithm used being used to generate the domain names.

Worms. Worm activity profits from the DNS by offering an alternative to scanning the IPv6 address space. If a worm is equipped with an accurate domain name generator, the speed at which the worm is able to spread can be increased.

Spam. Methods to secure email often work via the DNS, for example: the Sender Policy Framework (SPF) facilitates allowlisting of mail servers via a `TXT` record; DomainKeys Identified Mail (DKIM) allows the receiving party to validate the signature with a public key retrieved from a `TXT` record. Large email providers treat email with correct SPF and DKIM setups differently than those without SPF and DKIM. Failing to pass these checks raises suspicion. These defenses solve the email forging problem. However, it does little for the spam problem, since spammers use these defenses themselves [219, 220]. Nowadays spam comes from domains with correct SPF and DKIM setups, which makes emails originating from these domains appear legitimate, and increases the chance it will be delivered. Hence, the DNS exacerbates spam by increasing the appearance of legitimacy.

When relying on the DNS, spammers however also create usage patterns useful for detection. Snowshoe spam domains, a type of spam which spreads the sending of spam over hundreds of hosts, is detectable via active DNS measurements. The prerequisite of this is that these domains should use SPF, because that forces them to register a domain and create a record for each email-sending host. With snowshoe spam there are hundreds of sending hosts, and each requires a record. Such configurations are abnormal. Using active DNS mea-

surements and a machine learning classifier, trained to recognize these kinds of domain configurations, snowshoe spam domains can be detected [12]. More specifically, the method proposed in [12] detects such domains 2 to 104 days before traditional blocklists.

8.4. Open Challenges

All of these attacks remain open challenges as they make use of the DNS in ways that cannot be prevented without deteriorating normal operation of the DNS. We have discussed how such attacks can be detected, but detection does not solve the challenge of abuse. Most of these challenges stem from the clash between the legacy of the DNS and its modern usage. It is difficult to predict the future, especially predicting malicious use.

Spoofing DNS requests. Since the DNS mainly runs over UDP (a connectionless protocol) attackers are able to spoof DNS requests to send traffic to their victim. Running DNS over TCP solves this 'problem'. However, forcing each DNS resolution to set up a TCP connection results in higher latencies and more load on the resolvers. Unintentionally, DoT and DoH solve this problem of spoofing by using handshake-based transport protocols. DNS *cookies* [221] are a lightweight protection against spoofing DNS requests but to be effective, they need to be widely deployed.

Who is allowed to use the DNS?. The DNS serves the queries of everyone. This means, on the one hand, that you can make use of the DNS with any equipment you have. But, on the other hand, this also means that malicious uses are not obstructed. As we have discussed, attackers may use the DNS as a covert communication channel as the DNS usually passes through network filters. Designing and implementing access controls into the current DNS will be difficult and tiresome. We think the solution to communicative abuse lies outside of the realm of the DNS.

Encrypted DNS. Encrypted DNS will become a problem for the detection of abuse. In many of the attacks we have discussed above, passive DNS measurements were used to detect the abuse. This is no longer an effective methodology once the DNS queries are encrypted. Detection methodologies based on active DNS measurements remain unaffected as queried data remains the same.

9. Open Challenges and Non-DNS Naming Systems

In the preceding parts of this article (Table 1), we have discussed the challenges which the DNS faces and introduced solutions. However, as we showed at the end of each section, open challenges remain and we list them again in Table 10. Before concluding this tutorial on DNS we make a short excursion into non-DNS naming systems. We look at the open challenges and discuss how other naming systems deal with similar challenges. The goal is to learn how we could address these challenges in the DNS. We discuss open challenges on the basis of two experimental naming systems of non-IP-Based internets and the concept of alternative naming systems based on blockchain technology. An overview of past and present experimental internetworking systems is available in [222].

9.1. RAINS

RAINS is the naming system of the SCION internet architecture ("Scalability, Control, and Isolation On Next-generation networks") [223] and is conceptually similar to the DNS. RAINS maps domain names to a tuple consisting of a so-called Isolation Domain (ISD), an AS, and an optional network address within the AS (IP or following a different addressing scheme). An ISD is the key concept of SCION and consists of a group of AS-es that have agreed on a set of common policies, such as the same legal regime or a set of cryptographic keys. An ISD has two tasks: (1) isolate faults and security incidents (e.g. routing hijacks and compromises of certificate authorities) to the ISD where they take place and (2) act as a root of trust, for instance to generate and issue certificates for users connecting to AS-es in the ISD.

RAINS is similar to the DNS in that it uses the same DNS-style domain names and is a hierarchical system that allows operators to delegate authority over parts of the name space to others (see Section 2.2.1). RAINS protocol messages consist of queries and signed assertions. Naming authorities publish names and their properties through zone files, which consist of so-called "assertions" (e.g. the ISD, AS, and network address of a domain name). Clients interact with query services (similar to validating recursive resolvers in the DNS) to obtain validated assertions, while name space authorities publish them through the same protocol using authority services (similar to authoritative servers).

9.2. Named Data Networking (NDN) and NDN DNS (NDNS)

Named Data Networking (NDN) [224] is an internetworking architecture that uses the paradigm of data-centric networking rather than traditional host-based communications. This means that NDN applications submit the name of a content item they want to access to the network (e.g. `/company/application/user/content`), requesting it to fetch it from any of the sites where the content item resides. As a result, NDN applications do not first need to look up the address of a remote host (e.g. using the DNS or RAINS). NDN's naming functions do therefore not provide a name-to-address mapping (the concept of an address does not even exist in NDN) and instead use names to forward NDN packets and to obtain security-related information about names [225].

To obtain a content item, an NDN consumer application sends a so-called "interest packet" to an NDN router. NDN routers send an interest packet to several upstream routers simultaneously, enabling any producer that has the content to respond. NDN content producers announce name prefixes, which NDN routers distribute through the network, similar to how BGP distributes network prefix advertisements. A producer satisfies an interest packet if it has the requested content item, which the NDN network returns to the consumer in a data packet.

Afanasyev et al. [226] introduced NDN DNS (NDNS) which implements some aspects of the DNS into the NDN architecture. NDN namespaces implementing NDNS are able to resolve NDN-like queries via NDN. By running on top of NDN, NDNS takes advantage of all built-in NDN features. Additionally, by signing all records NDNS gains a hierarchical trust model similar to DNSSEC.

9.3. Blockchain-based Naming

Besides implementing new naming spaces in alternative Internet infrastructures, several efforts are made to develop a naming system for today's Internet based on append-only hash chains, called blockchains. Instead of a hierarchical naming system like the DNS, blockchain based naming systems are distributed between the peers of the name space, where every peer keeps every record of the name space locally. Peers can add new names, and only the owner of a name can release it again. Well-known implementations are Namecoin [227], Emercoin [228], or Blockstack [229], but none of these are deployed at a scale remotely comparable to DNS.

In theory, clients retrieving information from a blockchain-based naming systems can keep a copy of the complete name space locally. This implies that queries are by their nature confidential. In reality, however, clients rely on a system accessing information on blockchain, comparable to resolvers in DNS [230]. The connection between the client and this system needs to be protected separately. Names in the blockchain are not confidential by design. Integrity, on the other side, is guaranteed and peers can verify if the information attached to a name is correct. Additional measures are required only if an intermediary system is used to retrieve information.

Table 10: Open challenges in the DNS

		Non-DNS naming system		
	DNS Challenge	RAINS	NDN	Blockchain based
Confidentiality	Resolver discovery	✗	N/A	✓/✗
	Query confidentiality at resolvers	✗	N/A	✗
	Encryption to Authoritatives	✓	✗	✓/✗
Integrity	Error prone deployment and maintenance	✓	✓	✗
	Client-side validation	✓	✓	✓
	Use of quantum-safe signing algorithms	✗	✓/✗	✗
Availability	DoS attacks against the system	✓	✗	✓
	Centralization of resolvers	✓	✓	✓
	Realistic monitoring vantage points	✗	✗	✗
Abuse	Query spoofing	✓	✓	✓/✗
	Access controls	✗	✗	✗
	Observable behaviour	✗	✓	✓/✗

9.4. Open Challenges

Table 10 gives an overview of the open challenges we have discussed in this paper and, if applicable, notes if RAINS, NDN, or blockchain-based naming systems solve the challenge (✓) or not (✗). The aim of this section is to learn from other technologies rather than advocate that we should migrate to these other systems.

Confidentiality. None of the described technologies address the challenge of resolver discovery. Only blockchain naming systems can function without resolvers and thus circumvent the challenge altogether. In practice, also these systems rely on resolvers for performance and scalability reasons.

One way to address the challenge of query-confidentiality at resolvers in DNS could be borrowed from NDN. Here, a client broadcasts that it wants a certain piece of data, but producers who have this data only see the next hop. Therefore,

'resolvers' – in the broadest sense of the word – can see the query contents but do not observe where the request for data is coming from. Applying the same principle to DNS, we could employ a cascade of forwarding resolvers. Then, a client sends the query to a random resolver, which forwards the query to another random resolver, and so on and so forth. Each resolver would only see parts of the client's queries. A simplified version of this approach is currently under standardisation in the IETF [107].

Instead of applying multiple resolvers, clients could employ their own resolver locally, similar to blockchain naming systems. Here, however, the client could not benefit from shared caches which could significantly reduce performance. Also, authoritative name servers of popular zones would still see large parts of the client's queries.

The connection between resolvers and authoritative name servers is addressed by RAINS by design. Here, both the path from client to resolver, as well as the path between resolver and authoritative name servers is encrypted since SCION uses TLS. It is unclear if this solutions scales when busy authoritative name servers receive queries from millions of client and thus also need to manage millions of TLS sessions simultaneously.

Blockchain naming systems avoid this issue, since in principle the complete names space is stored directly at the client. In DNS, this is not possible since the name space of the DNS is not public – resolvers thus can never have the complete picture.

Integrity. First, the integrity assurances offered by DNSSEC are optional and must be deployed by DNS operators. Deployment rates of DNSSEC are rather low due to the complexity of DNSSEC. RAINS and NDN implement message integrity *by design*. This reduces complexity and does not require additional efforts by operators. By enabling DNSSEC signing and validation in DNS software by default and by automating as many processes as possible (e.g. signing and key rollovers), the complexity of DNSSEC can be reduced. This is as close as possible as we can come to fully embedding message integrity in DNS.

Also, clients rarely validate DNSSEC signatures themselves but trust resolvers to do this on their behalf. RAINS and NDN clients, again, have embedded client-side validation in the protocol design, providing end-to-end protection. Something similar could be achieved if encrpyted DNS becomes implemented in end user applications. At the moment, neither Chrome nor Firefox have plans to do so.

Last, as pointed out, quantum-computers have the potential to break all signing algorithms. This is also true for RAINS, NDN(S) and blockchain-based naming systems. RAINS, however, relies on TLS and the first large scale pilots have shown that quantum-safe algorithms can probably be applied in TLS without too much additional effort [231]. Fully moving DNS to DoH or DoT would not address this challenge since both do not provide the same protection as DNSSEC.

Availability. We have seen DoS and DDoS attacks against the DNS disrupting large parts of the Internet. SCION is designed with built-in DDoS defenses, like source authentication and by-design multipath communication [223]. Therefore, the naming service RAINS is resilient to such attacks as well.

The blockchain-based naming systems combine decentralization with DoS protection. By distributing the entire (naming) blockchain to each node there is no single point of failure. Similarly, with RFC 8806 [232], the zone file of the DNS root zone can also be distributed across each resolver, reducing the impact if the root servers become unavailable. Allman even goes further and discusses eliminating the root servers altogether [233].

We also raised concerns about the centralization of the DNS to a handful of large providers. None of the other naming systems share this challenge. E.g., we think that in an architecture like SCION there will be at least a few resolvers per ISD, as each name resolution is specific to its ISD context. In NDN data is named directly. Because of this we expect that if name-space owners would like to deploy NDNs, the resolver would become part of their own infrastructure, rather than a centralized service offered by a large company. Also, blockchain-based naming systems often boast that there is no central entity capable of taking down domains, because of the decentralization goal. In [234], Schomp et al. argue for leaving recursive resolving to the clients. This would make recursive resolving fully decentralized, leaving only the possibility of centralized authoritative name servers. To our knowledge, this proposal has not gained much traction yet.

As outlined in Section 7 monitoring the DNS for outages or badly-performing instances from the perspective of clients is difficult. Unfortunately, RAINS, NDN(S), and blockchain-based naming systems all suffer the same challenge.

Abuse. Query spoofing in the DNS may be misused to perform DDoS attacks. Spoofing is possible since the DNS uses a connectionless protocol.

RAINS, in the experimental setup, runs over a TLS connection. Hence, spoofing of RAINS queries is not possible. Also here, DoH and DoT can address this issue in DNS, by relying on spoofing-resistant TCP connections.

In Section 8 we talked about the challenge of access control. If abusive use of the DNS is detected there is no way of denying access to this user within the system. The alternatives discussed here face the same challenge. Access controls have been implemented in the NDN system, so theoretically NDNS can make use of this to control access. However, the method is manageable only for small sets of users. Essentially, it follows an allowlisting approach – no one has access unless you are listed. For a public naming system such a method is not scalable. We are still of the opinion that a naming system should not be policing who has access and who does not.

Last, encryption hinders the visibility of DNS abuse that operators have. NDN does not face the same challenge, since lookups are not encrypted. This, however, is not desirable in most DNS deployments. Early work shows [235, 236] that DoH and DoT do not provide perfect protection against eavesdropping, which partially restores the possibility for operators to legitimately observe DNS traffic. On the other hand, this also means that the challenge of confidentiality is still not addressed fully.

10. Summary

In this paper we have provided the first comprehensive tutorial on the DNS. We covered the basic functioning of the DNS, its deployment, and how the DNS can be measured to support DNS-based research. We then look at confidentiality, integrity, availability and abuse in the DNS, highlighting the *real world* challenges existing in modern deployments. For each of these, we identified a number of open challenges, pointing the reader to areas that still require additional research. Finally, we discussed how non-DNS naming systems address these open challenges and whether their approaches can be applied to the DNS as well.

Writing this paper has made use realize that, despite already being more than thirty years old, the DNS is still evolving rapidly. For example, important technical developments, in particular the introduction of DNS-over-HTTPS, are subject to heated debate in the community. For this reason we believe that the information provided in this paper can be of help to students and practitioners alike that want to deepen their understanding of the DNS or undertake research in this field.

Acknowledgments

We would like to thank Tom Grolleman and Marc Groeneweg for supplying data and reviewing our manuscript. This work has been funded by SIDN Fonds, an independent fund on the initiative of SIDN, the registrar for '.nl' domains. This work has partially been funded by the EU H2020 CONCORDIA (#830927) project, a Cybersecurity Competence Network. Additionally funded by the Open Technology Fund, a fund supporting Internet freedom worldwide. And the work has been partially funded by Salesforce.

References

[1] P. Mockapetris, R. White, D. Sharp, J. Martin, History of Networking (podcast) - Origins of the DNS (2018).
URL `https://networkcollective.com/2018/01/hon-dns-origins/`

[2] P. Vixie, R. White, D. Sharp, E. Sharp, History of Networking (podcast) - DNS Adoption (2018).
URL `https://networkcollective.com/2018/01/dns-adoption/`

[3] T. April, L. Chapin, k. claffy, C. Hesselman, M. Kaeo, J. Latour, D. McPherson, D. Piscitello, R. Rasmussen, M. Seiden, The DNS and the Internet of Things: Opportunities, Risks, and Challenges (2019).
URL `https://www.icann.org/news/blog/dns-and-the-internet-of-things-opportunities-risks-and-challenges`

[4] S. Bortzmeyer, DNS Privacy Considerations, RFC 7626, RFC Editor (2015).
URL `https://tools.ietf.org/html/rfc7626`

[5] D. Kaminsky, Black Ops 2008: It's the End of the Cache As We Know It, in: Black Hat USA, 2008.

[6] S. Hilton, Dyn Analysis Summary Of Friday October 21 Attack. (2016).
URL `https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/`

[7] J. Klensin, DNS Privacy, Authorization, Special Uses, Encoding, Characters, Matching, and Root Structure: Time for Another Look?, RFC 8324, RFC Editor (2018).
URL `https://tools.ietf.org/html/rfc8324`

[8] S. M. Bellovin, Using the Domain Name System for System Break-ins., in: Proceedings of the 1995 conference on USENIX Security Symposium, 1995.

[9] R. van Rijswijk-Deij, M. Jonker, A. Sperotto, A. Pras, The Internet of Names: A DNS Big Dataset Actively Measuring 50% of the Entire DNS Name Space, Every Day, in: Proceedings of ACM SIGCOMM 2015, ACM

Press, London, UK, 2015, pp. 91–92. doi:10.1145/2785956.2789996.

[10] R. van Rijswijk-Deij, M. Jonker, A. Sperotto, A. Pras, A High-Performance, Scalable Infrastructure for Large-Scale Active DNS Measurements, IEEE Journal on Selected Areas in Communications (2016).

[11] M. Müller, G. C. M. Moura, R. de O. Schmidt, J. Heidemann, Recursives in the Wild: Engineering Authoritative DNS Servers, in: Proceedings of the 2017 ACM Internet Measurement Conference, 2017.

[12] O. van der Toorn, R. van Rijswijk-Deij, A. Sperotto, Melting the Snow: Using Active DNS Measurements to Detect Snowshoe Spam Domains, in: Proceedings of the 2018 IEEE/IFIP Network Operations and Management Symposium, 2018.

[13] Internet Systems Consortium, DNS RFC (2017).
URL https://www.isc.org/community/rfcs/dns/

[14] B. Rampling, D. Dalan, DNS for Dummies, For Dummies, 2003.

[15] C. Liu, P. Albitz, DNS and Bind, " O'Reilly Media, Inc.", 2006.

[16] B. Hubert, Hello DNS (2018).
URL https://github.com/ahupowerdns/hello-dns/

[17] T. H. Kim, D. Reeves, A Survey of Domain Name System Vulnerabilities and Attacks, Journal of Surveillance, Security and Safety 1 (1) (2020) 34–60.

[18] A. Khormali, J. Park, H. Alasmary, A. Anwar, M. Saad, D. Mohaisen, Domain Name System Security and Privacy: A Contemporary Survey, Computer Networks 185 (2021) 107699. doi:https://doi.org/10.1016/j.comnet.2020.107699.
URL https://www.sciencedirect.com/science/article/pii/S1389128620313001

[19] R. Chandramouli, S. Rose, Challenges in Securing the Domain Name System, IEEE Security & Privacy 4 (1) (2006) 84–87.

[20] F. Zou, S. Zhang, B. Pei, L. Pan, L. Li, J. Li, Survey on Domain Name System Security, in: 2016 IEEE First International Conference on Data Science in Cyberspace (DSC), 2016, pp. 602–607. doi:10.1109/DSC.2016.96.

[21] A. Ramdas, R. Muthukrishnan, A survey on dns security issues and mitigation techniques, in: 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 781–784. doi:10.1109/ICCS45141.2019.9065354.

[22] N. Usman Aijaz, M. Misbahuddin, S. Raziuddin, Survey on dns-specific security issues and solution approaches, in: D. S. Jat, S. Shukla, A. Unal, D. K. Mishra (Eds.), Data Science and Security, Springer Singapore, Singapore, 2021, pp. 79–89.

[23] M. Feily, A. Shahrestani, S. Ramadass, A survey of botnet and botnet detection, in: 2009 Third International Conference on Emerging Security Information, Systems and Technologies, IEEE, 2009, pp. 268–273.

[24] Y. Zhauniarovich, I. Khalil, T. Yu, M. Dacier, A survey on Malicious Domains Detection through DNS Data Analysis, ACM Computing Surveys (CSUR) 51 (4) (2018) 1–36.

[25] S. Torabi, A. Boukhtouta, C. Assi, M. Debbabi, Detecting Internet Abuse by Analyzing Passive DNS Traffic: A Survey of Implemented Systems, IEEE Communications Surveys Tutorials 20 (4) (2018) 3389–3415. doi:10.1109/COMST.2018.2849614.

[26] M. Khonji, Y. Iraqi, A. Jones, Phishing Detection: a Literature Survey, IEEE Communications Surveys & Tutorials 15 (4) (2013) 2091–2121.

[27] W. Stewart, Living Internet (2015).
URL http://www.livinginternet.com

[28] K. Harrenstien, M. Stahl, E. Feinler, DOD Internet Host Table Specification, RFC 952 (1985).
URL https://tools.ietf.org/html/rfc952

[29] P. Mockapetris, Domain Names - Concepts and Facilities, RFC 882, RFC Editor (1983).
URL https://tools.ietf.org/html/rfc882

[30] P. Mockapetris, Domain Names - Implementation and Specification, RFC 883, RFC Editor (1983).
URL https://tools.ietf.org/html/rfc883

[31] J. Postel, The Domain Names Plan and Schedule, RFC 881, RFC Editor (1983).
URL https://tools.ietf.org/html/rfc881

[32] J. Postel, Domain Name System Implementation Schedule, RFC 897, RFC Editor (1984).
URL https://tools.ietf.org/html/rfc897

[33] J. Postel, Domain Name System Implementation Schedule - Revised, RFC 921, RFC Editor (1984).
URL https://tools.ietf.org/html/rfc921

[34] P. Mockapetris, Domain Names - Concepts and Facilities, RFC 1034, RFC Editor (1987).
URL https://tools.ietf.org/html/rfc1034

[35] P. Mockapetris, Domain Names - Implementation and Specification, RFC 1035, RFC Editor (1987).
URL http://tools.ietf.org/html/rfc1035

[36] P. Hoffman, A. Sullivan, K. Fujiwara, DNS Terminology, RFC 8499, RFC Editor (2019).
URL https://tools.ietf.org/html/rfc8499

[37] International Organization for Standardisation, ISO 3166 - Country Codes.
URL http://www.iso.org/iso/home/standards/country_codes.htm

[38] ICANN, Registrar Accreditation Agreement (2013).
URL https://www.icann.org/resources/pages/approved-with-specs-2013-09-17-en

[39] S. Hollenbeck, Extensible Provisioning Protocol (EPP), RFC 5730, RFC Editor (2009).
URL https://tools.ietf.org/html/rfc5730

[40] ICANN Governmental Advisory Committee, New gTLD Subsequent Procedures Policy Development Process (2019).
URL https://gac.icann.org/briefing-materials/public/icann65-gac-briefing-04.1-newgtld-subsequent-procedures-v1-6jun19.pdf

[41] D. Eastlake, Domain Name System (DNS) IANA Considerations, RFC 6895, RFC Editor (2013).
URL https://tools.ietf.org/html/rfc6895

[42] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, P. Hoffman, Specification for DNS over Transport Layer Security (TLS), RFC 7858, RFC Editor (2016).
URL https://tools.ietf.org/html/rfc7858

[43] Y. Cheng, J. Chu, S. Radhakrishnan, A. Jain, TCP Fast Open, RFC 7413, RFC Editor (2014).
URL https://tools.ietf.org/html/rfc7413

[44] S. Kitterman, Sender Policy Framework (SPF) for Authorising Use of Domains in Email, Version 1, RFC 7208, RFC Editor (2014).
URL https://tools.ietf.org/html/rfc7208

[45] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, L. Zhang, Impact of Configuration Errors on DNS Robustness, in: Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Proto-

cols for Computer Communications, 2004.

[46] R. Sommese, G. C. M. Moura, M. Jonker, R. van Rijswijk-Deij, A. Dainotti, K. Claffy, A. Sperotto, When parents and children disagree: Diving into dns delegation inconsistency, in: Proceedings of the 2020 Passive and Active Measurements Conference, 2020, accepted for publication.

[47] P. Koch, M. Larson, P. Hoffman, Initializing a DNS Resolver with Priming Queries, RFC 8109, RFC Editor (2017).
URL https://tools.ietf.org/html/rfc8109

[48] P. Mockapetris, K. J. Dunlap, Development of the Domain Name System, in: Symposium Proceedings on Communications Architectures and Protocols, 1988, pp. 123–133.

[49] OpenDNS, More Than 1 Percent of the World's Internet Users Now Using OpenDNS for a Safer, Faster, Smarter and More Reliable Connection (2010).
URL https://web.archive.org/web/20100325022738/http://www.opendns.com/about/announcements/160/

[50] J. S. Otto, M. A. Sánchez, J. P. Rula, F. E. Bustamante, Content Delivery and the Natural Evolution of DNS: Remote DNS Trends, Performance Issues and Alternative Solutions, in: Proceedings of the 2012 Internet Measurement Conference, 2012, pp. 523–536.

[51] K. Schomp, T. Callahan, M. Rabinovich, M. Allman, On Measuring the Client-side DNS Infrastructure, in: Proceedings of the 2013 Conference on Internet Measurement Conference, 2013, pp. 77–90.

[52] W. B. de Vries, R. V. Rijswijk-Deij, P. de Boer, A. Pras, Passive Observations of a Large DNS Service: 2.5 Years in the Life of Google, in: Proceedings of the 2018 IFIP Network Traffic Measurement and Analysis Conference, 2018.

[53] APNIC Statisticts Website, Use of DNS Resolvers for World (XA), https://stats.labs.apnic.net/rvrs (2020).

[54] DNSThought website, Atlas measurements used with DNSThought, https://dnsthought.nlnetlabs.nl/raw/ (2018).

[55] Mozilla, What's next in making Encrypted DNS-over-HTTPS the Default (2019).
URL https://blog.mozilla.org/futurereleases/2019/09/06/whats-next-in-making-dns-over-https-the-default/

[56] Chromium Blog, Experimenting with same-provider DNS-over-HTTPS upgrade (2019).
URL https://blog.chromium.org/2019/09/experimenting-with-same-provider-dns.html?m=1

[57] T. Pauly, Apple WWDC 2020 - Enabling Encrypted DNS (2020).
URL https://developer.apple.com/videos/play/wwdc2020/10047/

[58] Microsoft Blog, Windows Insiders can now test DNS over HTTPS (2020).
URL https://techcommunity.microsoft.com/t5/networking-blog/windows-insiders-can-now-test-dns-over-https/ba-p/1381282

[59] C. A. Shue, A. J. Kalafut, M. Gupta, The Web is Smaller than it Seems, in: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, 2007, pp. 123–128.

[60] S. Hao, H. Wang, A. Stavrou, E. Smirni, On the DNS Deployment of Modern Web Services, in: 2015 IEEE 23rd International Conference on Network Protocols (ICNP), IEEE, 2015, pp. 100–110.

[61] Abhishta, R. van Rijswijk-Deij, L. J. Nieuwenhuis, Measuring the Impact of a Successful DDoS Attack on the Customer Behaviour of Managed DNS Service Providers, ACM SIGCOMM Computer Communication Review (CCR) 48 (5) (2018) 70–76.

[62] S. Bates, J. Bowers, S. Greenstein, J. Weinstock, Y. Xu, J. Zittrain, Evidence of Decreasing Internet Entropy: The Lack of Redundancy in DNS Resolution by Major Websites and Services, Tech. rep., National Bureau of Economic Research (2018).

[63] M. Allman, Comments On DNS Robustness, in: Proceedings of the 2018 ACM Internet Measurement Conference, 2018.

[64] R. Edmonds, dnstap website (2019).
URL http://dnstap.info

[65] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, P. Gill, et al., Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem, in: The 25th Annual Network and Distributed System Security Symposium (NDSS 2018), 2018.

[66] L. Bilge, E. Kirda, C. Kruegel, M. Balduzzi, Exposure: Finding malicious domains using passive dns analysis., in: Ndss, 2011, pp. 1–17.

[67] S. Castro, D. Wessels, M. Fomenkov, K. Claffy, A day at the root of the internet, ACM SIGCOMM Computer Communication Review 38 (5) (2008) 41–46.

[68] G. Moura, M. Müller, M. Davids, M. Wullink, C. Hesselman, Fragmentation, truncation, and timeouts: are large DNS messages falling to bits?, in: International Conference on Passive and Active Network Measurement, Springer, 2021, pp. 460–477.

[69] Maarten Wullink, Giovane C. M. Moura, Muller, M, and Cristian Hesselman, ENTRADA: a High Performance Network Traffic Data Streaming Warehouse, in: Proceedings of the 2016 IEEE/IFIP Network Operations and Management Symposium, 2016.

[70] A. Dulaunoy, A. Kaplan, P. Vixie, H. Stern, Passive DNS - Common Output Format (2017).
URL https://tools.ietf.org/id/draft-dulaunoy-dnsop-passive-dns-cof-03.html

[71] Internet System Consortium, Privacy Considerations for ISC Passive DNS (2012).
URL https://www.farsightsecurity.com/assets/media/download/passive-dns-privacy.pdf

[72] J. M. Spring, C. L. Huth, The Impact of Passive DNS Collection on End-user Privacy, in: Proceedings of the SATIN 2012 Workshop, Teddington, UK, 2012.

[73] O. van der Toorn, R. van Rijswijk-Deij, T. Fiebig, M. Lindorfer, A. Sperotto, TXTing 101: Finding Security Issues in the Long Tail of DNS TXT Records, in: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW), 2020.

[74] A. Kountouras, P. Kintis, C. Lever, Y. Chen, Y. Nadji, D. Dagon, M. Antonakakis, R. Joffe, Enabling Network Security Through Active DNS Datasets, in: Proceedings of the 2016 International Symposium Research in Attacks, Intrusions, and Defenses, 2016.

[75] Q. Scheitle, T. Chung, J. Hiller, O. Gasser, J. Naab, R. van Rijswijk-Deij, O. Hohlfeld, R. Holz, D. Choffnes, A. Mislove, et al., A first look at certification authority authorization (CAA), ACM SIGCOMM Computer Communication Review 48 (2) (2018) 10–23.

[76] A. Akhavan Niaki, W. Marczak, S. Farhoodi, A. McGregor, P. Gill, N. Weaver, Cache me outside: A new look at dns cache probing, in: O. Hohlfeld, A. Lutu, D. Levin (Eds.), Passive and Active Measurement, Springer International Publishing, Cham, 2021, pp. 427–443.

[77] J. Davis, C. T. Deccio, A peek into the dns cookie jar-an analysis of dns cookie use., in: PAM, 2021, pp. 302–316.

[78] P. Foremski, O. Gasser, G. Moura, DNS Observatory: The Big Picture of the DNS, in: Proceedings of the 2019 ACM Internet Measurement Conference 2019, 2019.

[79] M. Skwarek, M. Korczynski, W. Mazurczyk, A. Duda, Characterizing Vulnerability of DNS AXFR Transfers with Global-Scale Scanning, in: 2019 IEEE Security and Privacy Workshops (SPW), IEEE, 2019, pp. 193–198.

[80] Google, Introduction to Google Public DNS (2019).
URL https://developers.google.com/speed/public-dns/docs/intro

[81] Cloudflare, Setting Up 1.1.1.1 - Cloudflare Resolver (2019).
URL https://developers.cloudflare.com/1.1.1.1/setting-up-1.1.1.1/

[82] Quad9, Quad9 Frequently Asked Questions (2019).
URL https://quad9.net/faq/

[83] Encrypted DNS Deployment Initiative, Encrypted DNS Deployment Initiative.

[84] D. P. Project, DNS Privacy Daemon - Stubby (2020).
URL https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+Daemon+-+Stubby

[85] Apple, Apple API DNSSettings (2020).
URL https://developer.apple.com/documentation/networkextension/nednssettingsmanager

[86] P. Hoffman, P. McManus, DNS Queries over HTTPS (DoH), RFC 8484, RFC Editor (2018).
URL https://tools.ietf.org/html/rfc8484

[87] Mozilla, Firefox extends privacy and security of Canadian internet users with by-default DNS-over-HTTPS rollout in Canada (2021).
URL https://blog.mozilla.org/en/mozilla/news/firefox-by-default-dns-over-https-rollout-in-canada/

[88] Chromium Blog, A safer and more private browsing experience with Secure DNS (2020).
URL https://blog.chromium.org/2020/05/a-safer-and-more-private-browsing-DoH.html

[89] T. Reddy.K, D. Wing, P. Patil, DNS over Datagram Transport Layer Security (DTLS), RFC 8094 (Feb. 2017). doi:10.17487/RFC8094.
URL https://rfc-editor.org/rfc/rfc8094.txt

[90] C. Huitema, A. Mankin, S. Dickinson, Specification of DNS over Dedicated QUIC Connections, Internet Draft (2020).
URL https://tools.ietf.org/html/draft-huitema-dprive-dnsoquic-00

[91] M. Bishop, Hypertext Transfer Protocol Version 3 (HTTP/3), Internet Draft (2020).
URL https://tools.ietf.org/html/draft-ietf-quic-http-32

[92] S. Bortzmeyer, DNS Query Name Minimisation to Improve Privacy, RFC 7816, RFC Editor (2016).
URL https://tools.ietf.org/html/rfc7816

[93] W. De Vries, Q. Scheitle, M. Müller, W. Toorop, R. Dolmans, R. Van Rijswijk-Deij, A First Look at QNAME Minimization in the Domain Name System, in: Proceedings of the 2019 Passive and Active Measurement Workshop, 2019.

[94] C. Contavalli, W. van der Gaast, D. Lawrence, W. Kumari, Client Subnet in DNS Queries, RFC 7871, RFC Editor (2016).
URL https://tools.ietf.org/html/rfc7871

[95] European Commission, EU data protection rules (2019).
URL https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en

[96] M. Kan, FTC Wants More Info on How ISPs Handle Your Data (2019).
URL https://www.pcmag.com/news/367429/ftc-wants-more-info-on-how-isps-handle-your-data

[97] J. Xu, J. Fan, M. H. Ammar, S. B. Moon, Prefix-preserving ip address anonymization: measurement-based security evaluation and a new cryptography-based scheme, in: Proceedings of the 2002 Conference on Network Protocol, IEEE, 2002, pp. 280–289.

[98] R. Pang, M. Allman, V. Paxson, J. Lee, The Devil and Packet Trace Anonymization, SIGCOMM Comput. Commun. Rev. (2006).

[99] Google, Your Privacy (2019).
URL https://developers.google.com/speed/public-dns/privacy

[100] Cloudflare, Privacy - Cloudflare Resolver (2019).
URL https://developers.cloudflare.com/1.1.1.1/commitment-to-privacy/

[101] Quad9, Privacy, Data Collection and Use Policy (2019).
URL https://quad9.net/policy/

[102] S. Dickinson, B. Overeinder, R. van Rijswijk-Deij, A. Mankin, Recommendations for DNS Privacy Service Operators, RFC 8932 (2020).
URL https://tools.ietf.org/html/rfc8932

[103] Mozilla, Security/DOH-resolver-policy (2019).
URL https://wiki.mozilla.org/Security/DOH-resolver-policy

[104] D. P. Project, DNS Privacy Public Resolvers (2020).
URL https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+Public+Resolvers

[105] Adaptive DNS Discovery Working Group (IEFT), ADD WG charter (2020).
URL https://datatracker.ietf.org/wg/add/about/

[106] DNS Privacu Working Group (IEFT), Oblivious DNS: Practical Privacy for DNS Queries, Proceedings on Privacy Enhancing Technologies (2020).
URL https://odns.cs.princeton.edu/pdf/pets.pdf

[107] E. Kinnear, P. McManus, T. Pauly, C. Wood, Oblivious DNS Over HTTPS, Internet Draft (2020).
URL https://tools.ietf.org/html/draft-pauly-dprive-oblivious-doh-03

[108] T. Verma, S. Singanamalla, Improving DNS Privacy with Oblivious DoH in 1.1.1.1 (2020).
URL https://blog.cloudflare.com/oblivious-dns/

[109] DNS Privacu Working Group (IEFT), DPRICVE WG charter (2020).
URL https://datatracker.ietf.org/wg/dprive/about/

[110] R. Zakon, Hobbes' Internet Timeline, RFC 2235, RFC Editor (1997).

URL https://tools.ietf.org/html/rfc2235
[111] A. Herzberg, H. Shulman, Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all. org, in: 2013 IEEE Conference on Communications and Network Security (CNS), IEEE, 2013, pp. 224–232.
[112] K. Man, Z. Qian, Z. Wang, X. Zheng, Y. Huang, H. Duan, DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels, in: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 1337–1350.
[113] ICANN Security and Stability Advisory Committee, SSAC Advisory on Registrant Protection: Best Practices for Preserving Security and Stability in the Credential Management Lifecycle (2015).
URL https://www.icann.org/en/system/files/files/sac-074-en.pdf
[114] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, DNS Security Introduction and Requirements, RFC 4033, RFC Editor (2005).
URL https://tools.ietf.org/html/rfc4033
[115] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, Resource Records for the DNS Security Extensions, RFC 4034, RFC Editor (2005).
URL https://tools.ietf.org/html/rfc4034
[116] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, Protocol Modifications for the DNS Security Extensions, RFC 4035, RFC Editor (2005).
URL https://tools.ietf.org/html/rfc4035
[117] J. Damas, M. Graff, P. Vixie, Extension Mechanisms for DNS (EDNS0), RFC 6891, RFC Editor (2013).
URL https://tools.ietf.org/html/rfc6891
[118] K. Moriarty, B. Kaliski, J. Jonsson, A. Rusch, PKCS #1: RSA Cryptography Specifications Version 2.2, RFC 8017, RFC Editor (2016).
URL https://tools.ietf.org/html/rfc8017
[119] B. Laurie, G. Sisson, R. Arends, D. Blacka, DNS Security (DNSSEC) Hashed Authenticated Denial of Existence, RFC 5155, RFC Editor (2008).
URL https://tools.ietf.org/html/rfc5155
[120] O. Kolkman, W. Mekking, R. Gieben, DNSSEC Operational Practices, Version 2, RFC 6781, RFC Editor (2012).
URL https://tools.ietf.org/html/rfc6781
[121] S. Morris, J. Ihren, J. Dickinson, W. Mekking, DNSSEC Key Rollover Timing Considerations, RFC 7583, RFC Editor (2015).
URL https://tools.ietf.org/html/rfc7583
[122] M. StJohns, Automated Updates of DNS Security (DNSSEC) Trust Anchors, RFC 5011, RFC Editor (2007).
URL https://tools.ietf.org/html/rfc5011
[123] ICANN, Update on the Root KSK Rollover Project (2017).
URL https://www.icann.org/news/blog/update-on-the-root-ksk-rollover-project
[124] D. Wessels, W. Kumari, P. Hoffman, Signaling Trust Anchor Knowledge in DNS Security Extensions (DNSSEC), RFC 8145, RFC Editor (2017).
URL https://tools.ietf.org/html/rfc8145
[125] M. Müller, M. Thomas, D. Wessels, W. Hardacker, T. Chung, W. Toorop, R. van Rijswijk-Deij, Roll, Roll, Roll your Root: A Comprehensive Analysis of the First Ever DNSSEC Root KSK Rollover, in: Proceedings of the 2019 ACM Internet Measurement Conference, 2019.
[126] ICANN, TLD DNSSEC Report (2019).
URL http://stats.research.icann.org/dns/tld_report/
[127] R. Lamb, DNSSEC Deployment Report (2019).
URL http://rick.eng.br/dnssecstat/
[128] Huston, Geoff, The State of DNSSEC Validation (2019).
URL https://blog.apnic.net/2019/03/14/the-state-of-dnssec-validation/
[129] N. Biasini, J. Esler, Threat Spotlight: Angler Lurking in the Domain Shadows (2015).
URL https://blogs.cisco.com/security/talos/angler-domain-shadowing
[130] FireEye, Global DNS Hijacking Campaign: DNS Record Manipulation at Scale (2019).
URL https://www.fireeye.com/blog/threat-research/2019/01/global-dns-hijacking-campaign-dns-record-manipulation-at-scale.html
[131] M. Korczyński, M. Król, M. van Eeten, Zone Poisoning: The How and Where of Non-Secure DNS Dynamic Updates, in: Proceedings of the 2016 ACM Internet Measurement Conference, 2016.
[132] C. Deccio, Maintenance, Mishaps and Mending in Deployments of the Domain Name System Security Extensions (DNSSEC), International Journal of Critical Infrastructure Protection 5 (2) (2012) 98–103.
[133] T. Chung, R. van Rijswijk-Deij, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, C. Wilson, A Longitudinal, End-to-End View of the {DNSSEC} Ecosystem, in: 26th {USENIX} Security Symposium ({USENIX} Security 17), 2017, pp. 1307–1322.
[134] M. Müller, T. Chung, A. Mislove, R. van Rijswijk-Deij, Rolling with Confidence: Managing the Complexity of DNSSEC Operations, IEEE Transactions on Network and Service Management (2019).
[135] G. Huston, J. da Silva Damas, W. A. Kumari, A Root Key Trust Anchor Sentinel for DNSSEC, RFC 8509 (Dec. 2018). doi:10.17487/RFC8509.
URL https://rfc-editor.org/rfc/rfc8509.txt
[136] W. A. Kumari, E. Hunt, R. Arends, W. Hardaker, D. C. Lawrence, Extended DNS Errors, RFC 8914 (Oct. 2020). doi:10.17487/RFC8914.
URL https://rfc-editor.org/rfc/rfc8914.txt
[137] Bugzilla, Browser-side validation of DNSSEC information (2019).
URL https://bugzilla.mozilla.org/show_bug.cgi?id=589538
[138] F. Arute, K. Arya, R. e. a. Babbush, Quantum Supremacy Using a Programmable Superconducting Processor, Nature 574 (2019) 505–510.
[139] P. W. Shor, Polynomial Time Algorithms for Discrete Logarithms and Factoring on a Quantum Computer, in: International Algorithmic Number Theory Symposium, Springer, 1994, pp. 289–289.
[140] M. Müller, W. Toorop, T. Chung, J. Janssen, R. van Rijswijk-Deij, The Reality of Algorithm Agility: Studying the DNSSEC Algorithm Life-Cycle, in: Proceedings of the 2020 ACM Internet Measurement Conference, 2020.
[141] M. Müller, J. de Jong, M. van Heesch, B. Overeinder, R. van Rijswijk-Deij, Retrofitting Post-Quantum Cryptography in Internet Protocols: a Case Study of DNSSEC, ACM SIGCOMM Computer Communication Review 50 (4) (2020) 49–57.

[142] W. Hardaker, Requirements for Management of Name Servers for the DNS, RFC 6168 (May 2011). doi:10.17487/RFC6168.
URL https://rfc-editor.org/rfc/rfc6168.txt
[143] R. Bush, D. Karrenberg, M. Kosters, R. Plzak, Root Name Server Operational Requirements, RFC 2870, RFC Editor (2000).
URL https://tools.ietf.org/html/rfc2870
[144] cz.nic, Knot DNS: Benchmark (2019).
URL https://www.knot-dns.cz/benchmark-old/
[145] ISC, BIND 9 Performance History (2017).
URL https://www.isc.org/blogs/bind9-performance-history/
[146] cz.nic, DNS Shotgun (2019).
URL https://gitlab.labs.nic.cz/knot/shotgun/
[147] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Z. Granville, A. Pras, Booters—An analysis of DDoS-as-a-service attacks, in: Proceedings of the 2015IFIP/IEEE International Symposium on Integrated Network Management, 2015.
[148] S. Kottler, February 28th DDoS Incident Report (2018).
URL https://github.blog/2018-03-01-ddos-incident-report/
[149] M. Jonker, A. King, J. Krupp, C. Rossow, A. Sperotto, A. Dainotti, Millions of targets under attack: a macroscopic characterization of the DoS ecosystem, in: Proceedings of the 2017 ACM Internet Measurement Conference, 2017.
[150] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin, N. Somaiya, Connection-oriented DNS to improve privacy and security, in: Proceedings of the 2015 IEEE Symposium on Security and Privacy, 2015.
[151] C. Kreibich, N. Weaver, B. Nechaev, V. Paxson, Netalyzr: Illuminating the Edge Network, in: Proceedings of the 2010 ACM SIGCOMM annual conference on Internet Measurement, 2010.
[152] G. van den Broek, R. M. van Rijswijk, A. Sperotto, A. Pras, DNSSEC Meets Real World: Dealing with Unreachability Caused by Fragmentation, IEEE Communications Magazine (2014).
[153] DNS Flag Day 2020 (2020).
URL https://dnsflagday.net/2020/
[154] J. Dickinson, S. Dickinson, R. Bellis, A. Mankin, D. Wessels, DNS Transport over TCP - Implementation Requirements, RFC 7766, RFC Editor (2016).
URL https://tools.ietf.org/html/rfc7766
[155] Y. Yu, D. Wessels, M. Larson, L. Zhang, Authority Server Selection in DNS Caching Resolvers, ACM SIGCOMM Computer Communication Review (2012).
[156] R. Elz, R. Bush, S. Bradner, M. Patton, Selection and Operation of Secondary DNS Servers, RFC 2182, RFC Editor (1997).
URL https://tools.ietf.org/html/rfc2182
[157] E. Lewis, A. H. (Ed.), DNS Zone Transfer Protocol (AXFR), RFC 5936, RFC Editor (2010).
URL https://tools.ietf.org/html/rfc5936
[158] M. Ohta, Incremental Zone Transfer in DNS, RFC 1995, RFC Editor (1996).
URL https://tools.ietf.org/html/rfc1995
[159] P. Vixie, A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY), RFC 1996, RFC Editor (1996).
URL https://tools.ietf.org/html/rfc1996
[160] P. Vixie, O. Gudmundsson, D. Eastlake 3rd, B. Wellington, Secret Key Transaction Authentication for DNS (TSIG), RFC 2845, RFC Editor (2000).
URL https://tools.ietf.org/html/rfc2845
[161] W. Toorop, S. Dickinson, S. K. Sahib, P. Aras, A. Mankin, DNS Zone Transfer over TLS, RFC 9103 (Aug. 2021). doi:10.17487/RFC9103.
URL https://rfc-editor.org/rfc/rfc9103.txt
[162] R. van Rijswijk-Deij, M. Jonker, A. Sperotto, A. Pras, The Internet of Names: A DNS Big Dataset, in: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, 2015.
[163] Alexa, Top 1M sites (2018).
URL https://www.alexa.com/topsites
[164] J. Abley, K. Lindqvist, Operation of Anycast Services, RFC 4786, RFC Editor (2006).
URL https://tools.ietf.org/html/rfc4786
[165] G. C. M. Moura, R. de O. Schmidt, J. Heidemann, W. B. de Vries, M. Müller, L. Wei, C. Hesselman, Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event, in: Proceedings of the 2016 ACM Internet Measurement Conference, 2016.
[166] W. B. De Vries, R. de O Schmidt, W. Hardaker, J. Heidemann, P.-T. de Boer, A. Pras, Broad and load-aware anycast mapping with verfploeter, in: Proceedings of the 2017 ACM Internet Measurement Conference, 2017.
[167] B. Quoitin, C. Pelsser, L. Swinnen, O. Bonaventure, S. Uhlig, Interdomain traffic engineering with BGP, IEEE Communications Magazine (5) (2003).
[168] P. Marcos, L. Prehn, L. Leal, A. Dainotti, A. Feldmann, M. Barcellos, AS-Path Prepending: There Is No Rose without a Thorn, in: Proceedings of the 2020 ACM Internet Measurement Conference, 2020.
[169] Root Server Operators, Root Server Technical Operations Assn (2017).
URL http://root-servers.org/
[170] R. Sommese, G. Akiwate, M. Jonker, G. C. Moura, M. Davids, R. van Rijswijk-Deij, G. M. Voelker, S. Savage, K. Claffy, A. Sperotto, Characterization of Anycast Adoption in the DNS Authoritative Infrastructure, in: Network Traffic Measurement and Analysis Conference (TMA'21), 2021.
[171] M. Davids, Local Anycasyt at SIDN, in: Proceedings of the 2018 CENTR RnD Workshop, 2018.
[172] O. Gudmundsson, Introducing DNS Resolver, 1.1.1.1 (not a joke) (2018).
URL https://blog.cloudflare.com/dns-resolver-1-1-1-1/
[173] DNS OARC, DSC - DNS Stats Collector (2019).
URL https://www.dns-oarc.net/tools/dsc
[174] G. C. M. Moura, J. Heidemann, M. Müller, R. de O. Schmidt, M. Davids, When the Dike Breaks: Dissecting DNS Defenses During DDoS, in: Proceedings of the 2018 ACM Internet Measurement Conference, 2018.
[175] D. Lawrence, W. Kumari, P. Sood, Serving Stale Data to Improve DNS Resiliency, RRC 8767 (2020).
URL https://tools.ietf.org/html/rfc8767
[176] ISC, Using the Response Rate Limiting Feature (2018).
URL https://kb.isc.org/docs/aa-00994
[177] R. van Rijswijk-Deij, A. Sperotto, A. Pras, DNSSEC and Its Potential for DDoS Attacks, in: Proceedings of the 2014 ACM Internet Measurement Conference, 2014.

[178] R. van Rijswijk-Deij, A. Sperotto, A. Pras, Making the Case for Elliptic Curves in DNSSEC, ACM SIGCOMM Computer Communication Review (2015).

[179] J. Abley, Ólafur Guðmundsson, M. Majkowski, E. Hunt, Providing Minimal-Sized Responses to DNS Queries That Have QTYPE=ANY, RFC 8482, RFC Editor (2019).
URL https://tools.ietf.org/html/rfc8482

[180] M. Kührer, T. Hupperich, J. Bushart, C. Rossow, T. Holz, Going wild: Large-scale classification of open DNS resolvers, in: Proceedings of the 2015 ACM Internet Measurement Conference, 2015.

[181] W. Kumari, P. Hoffman, Decreasing Access Time to Root Servers by Running One on Loopback, RFC 7706 (Informational) (Nov. 2015). doi:10.17487/RFC7706.
URL https://www.rfc-editor.org/rfc/rfc7706.txt

[182] G. C. Moura, S. Castro, W. Hardaker, M. Wullink, C. Hesselman, Clouding up the Internet: how centralized is DNS traffic becoming?, in: Proceedings of the ACM Internet Measurement Conference, 2020, pp. 42–49.

[183] RIPE NCC Staff, RIPE Atlas: A Global Internet Measurement Network, Internet Protocol Journal (IPJ) 18 (3) (Sep 2015).

[184] G. C. M. Moura, J. Heidemann, W. Hardaker, J. Bulten, J. Ceron, C. Hesselman, Old but Gold: Prospecting TCP to Engineer DNS Anycast (extended), Tech. rep., USC/Information Sciences Institute (2020).
URL https://www.isi.edu/%7ejohnh/PAPERS/Moura20a.html

[185] L. Krämer, J. Krupp, D. Makita, T. Nishizoe, T. Koide, K. Yoshioka, C. Rossow, AmpPot: Monitoring and Defending Amplification DDoS Attacks, in: Proceedings of the 2015 International Symposium on Research in Attacks, Intrusions and Defenses, 2015.

[186] J. Krupp, M. Backes, C. Rossow, Identifying the Scanners and Attack Infrastructure behind Amplification DDoS attacks, in: Proceedings of the 2016 ACM Conference on Computer and Communications Security, 2016.

[187] J. Krupp, M. Karami, C. Rossow, D. McCoy, M. Backes, Linking Amplification DDoS Attacks to Booter Services, in: Proceedings of the 2017 International Symposium on Research in Attacks, Intrusions and Defenses, 2017.

[188] A. Noroozian, M. Korczyński, C. H. Gañan, D. Makita, K. Yoshioka, van Michel Eeten, Who Gets the Boot? Analyzing Victimization by DDoS-as-a-Service, in: Research in Attacks, Intrusions, and Defenses, 2016.

[189] D. Thomas, R. Clayton, A. Beresford, 1000 days of UDP amplification DDoS attacks, in: Proceedings of the 2017 APWG Symposium on Electronic Crime Research, 2017.

[190] C. Fachkha, E. Bou-Harb, M. Debbabi, Fingerprinting Internet DNS Amplification DDoS Activities, in: Proceedings of the 2014 IFIP International Conference on New Technologies, Mobility and Security, 2014.

[191] M. Majkowski, O. Gudmundsson, Deprecating the DNS ANY meta-query type (2015).
URL https://blog.cloudflare.com/deprecating-dns-any-meta-query-type/

[192] A. Sperotto, O. van der Toorn, R. van Rijswijk-Deij, TIDE: Threat Identification Using Active DNS Measurements, in: Proceedings of the 2017 ACM SIGCOMM Posters and Demos, 2017.

[193] P. Kintis, N. Miramirkhani, C. Lever, Y. Chen, R. Romero-Gómez, N. Pitropakis, N. Nikiforakis, M. Antonakakis, Hiding in Plain Sight: A Longitudinal Study of Combosquatting Abuse, in: Proceedings of the 2017 ACM Special Interest Group on Security, Audit and Control, 2017.

[194] ICANN, Cybersquatting (2013).
URL https://www.icann.org/resources/pages/cybersquatting-2013-05-03-en

[195] O. van der Toorn, A. Sperotto, Looking beyond the horizon: Thoughts on Proactive Detection of Threats, ACM Journal on Digital Threats: Research and PracticeAccepted for publication (2019).

[196] Y.-M. Wang, D. Beck, J. Wang, C. Verbowski, B. Daniels, Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting, in: Proceedings of the 2006 Conference on Steps to Reducing Unwanted Traffic on the Internet, 2006.

[197] N. Nikiforakis, S. V. Acker, W. Meert, L. Desmet, F. Piessens, W. Joosen, Bitsquatting: exploiting bitflips for fun, or profit?, in: Proceedings of the 2013 International Conference on World Wide Web, 2013.

[198] S. Maroofi, M. Korczynski, A. Duda, From Defensive Registration to Subdomain Protection: Evaluation of Email Anti-Spoofing Schemes for High-Profile Domains, in: Proceedings of Network Traffic Measurement and Analysis Conference (TMA) 2020, 2020.

[199] N. Nikiforakis, M. Balduzzi, L. Desmet, F. Piessens, W. Joosen, Soundsquatting: Uncovering the Use of Homophones in Domain Squatting, in: Proceedings of the 2014 Information Security, 2014.

[200] H. Suzuki, D. Chiba, Y. Yoneya, T. Mori, S. Goto, ShamFinder: An Automated Framework for Detecting IDN Homographs, in: Proceedings of the 2019 ACM Internet Measurement Conference, 2019.

[201] R. Yazdani, O. van der Toorn, A. Sperotto, A Case of Identity: Detection of Suspicious IDN Homograph Domains Using Active DNS Measurements, in: Proceedings of the 5th International Workshop on Traffic Measurements for Cybersecurity (WTMC 2020), 2020.

[202] C. Li, W. Jiang, X. Zou, Botnet: Survey and Case Study, in: Proceedings of the 2009 International Conference on Innovative Computing, Information and Control, 2009.

[203] Cisco Talos Intelligence Group, Covert Channels and Poor Decisions: The Tale of DNSMessenger (2017).
URL https://blog.talosintelligence.com/2017/03/dnsmessenger.html

[204] Cisco Talos Intelligence Group, Spoofed SEC Emails Distribute Evolved DNSMessenger (2017).
URL https://blog.talosintelligence.com/2017/10/dnsmessenger-sec-campaign.html

[205] J. Kwon, J. Lee, H. Lee, A. Perrig, PsyBoG: A scalable botnet detection method for large-scale DNS traffic, Computer Networks (2016).

[206] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, E. Gerhards-Padilla, A Comprehensive Measurement Study of Domain Generating Malware, in: Proceedings of the 2016 USENIX Conference on Security Symposium, 2016.

[207] L. Asher-Dotan, What is domain generation algorithm: 8 real world dga variants (2016).
URL https://www.cybereason.com/blog/what-are-

domain-generation-algorithms-dga
[208] X. Hoang, Q. Nguyen, Botnet Detection Based On Machine Learning Techniques Using DNS Query Data, Future Internet (2018).
[209] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, D. Dagon, From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware, in: Proceedings of the 21st USENIX Conference on Security Symposium, 2012.
[210] J. Woodbridge, H. S. Anderson, A. Ahuja, D. Grant, Predicting Domain Generation Algorithms with Long Short-Term Memory Networks, CoRR abs/1611.00791 (2016).
URL http://arxiv.org/abs/1611.00791
[211] A. McNeil, How did the WannaCry ransomworm spread? (2017).
URL https://blog.malwarebytes.com/cybercrime/2017/05/how-did-wannacry-ransomworm-spread/
[212] A. Kamra, H. Feng, V. Misra, A. D. Keromytis, The effect of DNS delays on worm propagation in an IPv6 Internet, in: Proceedings of the 2005 IEEE International Conference on Computer and Information Technology, 2005.
[213] P. Kammas, T. Komninos, Y. C. Stamatiou, Modeling the Co-evololution DNS Worms and Anti-worms in IPv6 Networks, in: Proceedings of the 2009 International Conference on Information Assurance and Security, Vol. 2, 2009.
[214] D. Whyte, E. Kranakis, P. V. Oorschot, DNS-Based Detection of Scanning Worms in an Enterprise Network, in: Proceedings of the 2005 Network and Distributed System Security Symposium, 2005.
[215] M. Konte, N. Feamster, J. Jung, Dynamics of Online Scam Hosting Infrastructure, in: Proceedings of the 2009 Passive and Active Network Measurement Conference, 2009.
[216] R. Perdisci, I. Corona, G. Giacinto, Early Detection of Malicious Flux Networks via Large-Scale Passive DNS Traffic Analysis, IEEE Transactions on Dependable and Secure Computing (2012).
[217] S. Yadav, A. K. K. Reddy, A. N. Reddy, S. Ranjan, Detecting Algorithmically Generated Malicious Domain Names, in: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, 2010. doi:10.1145/1879141.1879148.
URL https://doi.org/10.1145/1879141.1879148
[218] E. Stalmans, B. Irwin, A Framework for DNS Based Detection and Mitigation of Malware Infections on a Network, in: Proceedings of the 2011 Information Security for South Africa, 2011.
[219] T. Zink, How Spammers Get Around SPF, Tech. rep., CircleID (2007).
URL http://www.circleid.com/posts/782012_spammer_get_around_spf/
[220] P. Roberts, Spammers use sender authentication too, study says, Tech. rep., ComputerWorld (2004).
URL https://www.computerworld.com/article/2566815/spammers-use-sender-authentication-too--study-says.html
[221] D. E. E. 3rd, M. P. Andrews, Domain Name System (DNS) Cookies, RFC 7873 (May 2016). doi:10.17487/RFC7873.
URL https://rfc-editor.org/rfc/rfc7873.txt
[222] D. D. Clark, Designing an Internet, MIT Press, 2018.
[223] A. Perrig, P. Szalachowski, R. M. Reischuk, L. Chuat, SCION: a secure Internet architecture, Springer, 2017.
[224] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang, et al., Named Data Networking, ACM SIGCOMM Computer Communication Review (2014).
[225] Z. Zhang, Y. Yu, H. Zhang, E. Newberry, S. Mastorakis, Y. Li, A. Afanasyev, L. Zhang, An Overview of Security Support in Named Data Networking, IEEE Communications Magazine (2018).
[226] A. Afanasyev, X. Jiang, Y. Yu, J. Tan, Y. Xia, A. Mankin, L. Zhang, NDNS: A DNS-Like Name Service for NDN, in: International Conference on Computer Communication and Networks, 2017.
[227] Namecoin, Namecoin (2019).
URL https://www.namecoin.org/
[228] Emercoin, EmerDNS (2019).
URL https://emercoin.com/en/documentation/blockchain-services/emerdns/emerdns-introduction
[229] M. Ali, J. Nelson, R. Shea, M. J. Freedman, Blockstack: A global naming and storage system secured by blockchains, in: Proceedings of the 2016 USENIX Annual Technical Conference, 2016.
[230] E. Karaarslan, E. Adiguzel, Blockchain Based DNS and PKI Solutions, IEEE Communications Standards Magazine (2018).
[231] K. Kwiatkowski, Towards Post-Quantum Cryptography in TLS, blog.cloudflare.com/towards-post-quantum-cryptography-in-tls/ (jun 2019).
[232] W. A. Kumari, P. E. Hoffman, Running a Root Server Local to a Resolver, RFC 8806 (Jun. 2020). doi:10.17487/RFC8806.
URL https://rfc-editor.org/rfc/rfc8806.txt
[233] M. Allman, On eliminating root nameservers from the dns, in: Proceedings of the 18th ACM Workshop on Hot Topics in Networks, 2019, pp. 1–8.
[234] K. Schomp, M. Allman, M. Rabinovich, Dns resolvers considered harmful, in: Proceedings of the 13th ACM Workshop on Hot Topics in Networks, 2014, pp. 1–7.
[235] Q. Huang, D. Chang, Z. Li, A comprehensive study of dns-over-https downgrade attack, in: 10th {USENIX} Workshop on Free and Open Communications on the Internet ({FOCI} 20), 2020.
[236] R. Houser, Z. Li, C. Cotton, H. Wang, An investigation on information leakage of dns over tls, in: Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies, 2019, pp. 123–137.

Acronyms

API Application Programming Interface. 17, 30

ARPANET Advanced Research Projects Agency Network. 3

AS Autonomous System. 37, 41

BGP Border Gateway Protocol. 31, 42

C&C Command and Control. 38–40

CDN Content Delivery Network. 40

DDoS Distributed Denial of Service. 11, 29, 31–36, 38, 39, 43, 44

DGA Domain Generation Algorithm. 39, 40

DKIM DomainKeys Identified Mail. 40

DNS Domain Name System. 1–44

DNSSEC DNS Security Extensions. 7, 8, 19, 21–27, 29, 30, 32, 43

DoH DNS-over-HTTPS. 17, 19, 29, 43, 44

DoS Denial of Service. 33, 34, 43

DoT DNS-over-TLS. 7, 16, 17, 19, 29, 43, 44

DTLS DNS over Datagram Transport Layer Security. 17

ECS EDNS Client Subnet. 18

EDNS0 Extension Mechanisms for DNS. 29

GDPR General Data Protection Regulation. 18

IANA Internet Assigned Numbers Authority. 5

ICANN Internet Corporation for Assigned Names and Numbers. 5, 27, 30, 37

IDN Internationalized Domain Name. 5, 37

IETF Internet Engineering Task Force. 2–4, 16, 20, 21, 26, 43

IP Internet Protocol. 1, 13, 29, 31, 35, 39, 40

ISD Isolation Domain. 41, 43

ISP Internet Service Provider. 2, 11, 12, 17–19, 21

KSK Key Signing Key. 23–27

MTU Maximum Transmission Unit. 29

NDN Named Data Networking. 42–44

RFC Request for Comments. 3

RRL Response Rate Limiting. 33

SPF Sender Policy Framework. 40

SSAC Security and Stability Advisory Committe. 27

TLD Top Level Domain. 30

TLS Transport Layer Security. 16, 17, 43, 44

TRR Trusted Recursive Resolver. 19

TTL Time To Live. 6, 8, 10, 13, 15, 40

UDRP Uniform Domain-Name Dispute Resolution Policy. 37

ZSK Zone Signing Key. 24–26