THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale: École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique

Spécialité: Computer Science

Unité de recherche: Laboratoire Informatique de Grenoble (LIG)

DNS as a Source of Trust and Information in the Modern Internet

Le DNS comme Source de Confiance et d'Information pour l'Internet Moderne

Présentée par :

Simon Fernandez

Direction de thèse :

Andrzej Duda
PROFESSEUR DES UNIVERSITES, Grenoble INP, Directeur de thèse

Maciej Korczyński
MAITRE DE CONFERENCES HDR, Grenoble INP, Co-Directeur de thèse

Rapporteurs :

Pascal Lafourcade
PROFESSEUR DES UNIVERSITES, Université Clermont Auvergne, Rapporteur

Gianluigi Ferrari
FULL PROFESSOR, Università di Parma, Rapporteur

Thèse soutenue publiquement le 5 Décembre 2023, devant le jury composé de :

Vivien Quema
PROFESSEUR DES UNIVERSITES, Grenoble INP - UGA, Président

Pascal Lafourcade
PROFESSEUR DES UNIVERSITES, Université Clermont Auvergne, Rapporteur

Gianluigi Ferrari
FULL PROFESSOR, Università di Parma, Rapporteur

Andrzej Duda
PROFESSEUR DES UNIVERSITES, Grenoble INP - UGA, Directeur de thèse

Invités :

Maciej Korczyński
MAITRE DE CONFERENCES HDR, Grenoble INP - UGA, Invité

It’s not DNS
There’s no way it’s DNS
It was DNS

SSBroski

"À celleux qui luttent"

Acknowledgments

I would like to thank my supervisors Maciej Korczyński and Andrzej Duda for their guidance during those three years. It has not always been simple and involved multiple late night rushes before deadlines, but I learned a lot from both of you, from how to write proper scientific english (even if this manuscript may be the proof that I still have a long road ahead of me) to how to become a researcher.

Thanks a lot to the DRAKKAR team and especially to Maud, Michele, Pascale, Arnaud, Frank, Olivier, Martin and Étienne for your hospitality, your technical and non-technical discussions over free-gliding, cooking, sewing, protesting and plumbing.

Special thanks to my thesis colleagues Yevheniya, Olivier, Jan, Ben, Sourena, Takwa, Josquin, Florent and Brandon, for the serious and less-serious talks, the restaurants, the shots, the strikes, the protests, the overall great atmosphere of the team and for being awesome road-companions during those years.

La suite des remerciements se fera en français car ils sont addressés à ma famille, mes proches et mes amis. Tout d'abord, un grand merci à mes parents, qui m'ont montré un chemin sans m'y pousser, qui m'ont accompagné, qui m'ont éclairé sur tant de choix qui m'ont mené à me lancer dans une thèse. Je suis vraiment heureux là où je suis, et je vous remercie du fond du coeur de m'avoir accompagné toutes ces années, et de continuer à m'accompagner au quotidien.

En suite, merci beaucoup aux membres de l'Auberge, vous êtes trop nombreuxes pour être toustes cité.es, mais je pense fort à vous. Vous êtes des gens formidables et vous m'avez si souvent tenu compagnie à distance, en étant une source sans fin d'interactions sociales, de bêtises et de soutien.

Une pensée toute spéciale pour Janelle, mais aussi Malo, Joël, Georges et Xavier. Je vous aime très fort, et je ne vous remercierai jamais assez pour tout ce qu'on a vécu ensemble. Chaque weekend avec vous a été une bouffée d'air frais qui me redonnait de l'énergie pour des semaines et me remplissait d'espoir.

Et enfin, un énorme merci à Julia. Tu es une personne formidable, et j'ai une chance inouïe de t'avoir rencontrée. Je ne sais absolument pas comment j'aurais pu passer ces années sans ton soutien permanent et sans ces moments ensemble qui me redonnent toujours le sourire et la bonne humeur.

Abstract

Abstract

The Domain Name System is a cornerstone of the modern Internet, providing information on millions of domain names by answering billions of requests per day. It is often simplified as just a system mapping human-readable names to machine IP addresses, but it fills multiple other roles and many systems use its architecture, availability, and resilience as a foundation for their design. The DNS is a hierarchical and distributed system, storing technical information on domain names, like their IP address allowing other devices on the network to contact them, or the name of the server in charge of their mail boxes. However, the DNS protocol was designed in the early 1980s, when the Internet was just a small set of interconnected universities and government agencies. Therefore, hundreds of protocol extensions were added to its specifications to better address the needs and paradigms of the growing and changing Internet. Thanks to its unique properties, many different systems rely on the domain name architecture and the DNS infrastructure, like mail delivery and security, load balancing, intrusion detection systems and service discovery. Malicious actors also leverage the DNS architecture to increase their reach, impact, or hide their identity, like spam campaigns, Denial of Service attacks, malware delivery or botnets control. Querying the DNS is often the first step of a connection between two devices on the Internet, so observing this traffic can reveal ongoing spam campaigns, software updates distributions, misconfigurations, the rise of the Internet of Things or cyberwars between countries. However, studying the DNS is a challenging task, considering the massive volume of queries and its distributed architecture. Nevertheless, observing even a sample of the DNS traffic can still provide crucial insights into how the modern Internet is organized and how different entities and systems use it.

In this thesis, we explore the DNS as a way to establish trust between entities on the Internet and as a source of information providing valuable insights on the Internet usages and diversity. We dive into the domain name registration process and challenge the assumption that the data stored in multiple domain name registration databases are always coherent. This hypothesis made by previous works turned out to be true for the majority of domains, but we raise concerns on some inconsistencies that still remain. We designed a naming scheme for constrained devices that leverages the DNS capabilities, allowing for efficient encoding of properties and location. This design also provides ways to discover devices based on these properties without adding load to the end devices. Finally, we study the security configuration and DNS traffic patterns of domains distributing spam. We propose a detection algorithm leveraging the differences between benign domains and spam domains to classify spam domains even before the start of a spam campaign, allowing defenders to take protective measures quickly and prevent attacks.

Résumé

Le Domain Name system est une pierre angulaire de l'Internet moderne, fournissant des informations techniques sur des millions de domaines en répondant à des milliards de requêtes chaque jour. Il est souvent réduit à sa fonction de table associant une adresse IP à chaque nom de domaine, mais le DNS a de nombreux autres rôles et un grand nombre de systèmes se basent sur son architecture et sa stabilité. Le DNS est un système hiérarchique distribué, associant des informations techniques à des noms de domaines, comme leur adresse IP sur le réseau permettant de les contacter, ou le nom du serveur en charge de la gestion de leurs mails. Cependant, le DNS a été créé au début des années 80, quand Internet n'était qu'un petit ensemble d'universités et agences gouvernementales interconnectées. Des centaines d'additions et extensions ont été ajoutées au protocole pour l'adapter aux besoins grandissants et changeants de l'Internet. Grâce à sa facilité d'utilisation, son adoption massive et son architecture résiliente, de nombreux systèmes s'appuient sur les noms de domaines et le DNS, comme les protocoles d'envoi et de réception de mails, les équilibreurs de charge, des systèmes de détection d'intrusion ou de découverte de services. Des systèmes mal intentionnés utilisent aussi l'architecture DNS pour augmenter leur efficacité ou cacher leur identité, comme l'envoi de pourriels, des attaques de déni de service, de la distribution de virus ou le contrôle de botnets. La majorité des connexions entre deux terminaux sur Internet commence par des requêtes DNS. Observer ces requêtes permet donc d'observer en direct de nombreux événements, comme des campagnes d'envoi de pourriels, le déploiement de mises à jour, des problèmes de configuration, la montée de l'Internet des Objets ou des cyber-conflits entre des états. Cependant, étudier le DNS est une tâche complèxe, étant donné l'important volume de trafic qu'il représente, et son architecture distribuée. Toutefois, même en se limitant à des échantillons du trafic réel, ce trafic permet de mieux comprendre comment l'Internet est organisé, et comment différents acteurs l'utilisent.

Dans ce travail de thèse, nous avons étudié le DNS dans son rôle d'établissement de liens de confiance entre terminaux et en tant que source d'information permettant de mieux comprendre la diversité et l'usage actuel d'Internet. Nous avons tout d'abord étudié le processus d'enregistrement de noms de domaines, en remettant en question une hypothèse, faites par plusieurs articles scientifiques et travaux techniques, que les multiples sources d'information sur les noms de domaines étaient toujours cohérentes entre elles. Nous avons apporté des preuves confirmant cette cohérence dans la majorité des cas, rassurant ainsi les travaux collectant ces données pour un grand nombre de domaines. Nous avons cependant souligné que certains types d'entrées étaient plus souvent erronés, et que les travaux se basant sur leur contenu doivent avoir une vigilance particulière vis à vis de leur cohérence. Nous avons créé un schéma de nommage pour des terminaux à capacités limitées, permettant d'encoder efficacement les propriétés et localisation du terminal. Ce système utilise l'infrastructure DNS et exploite le format des noms de domaines pour permettre des requêtes rapides et efficaces, n'impliquant pas de charge supplémentaire pour les terminaux découverts. Enfin, nous avons étudié les entrées DNS de configurations de sécurité et le trafic DNS des domaines envoyant des pourriels. Nous avons construit un outil de détection utilisant des différences de configuration entre domaines bénins et domaines malveillants pour détecter les domaines malveillants avant qu'ils n'envoient le moindre mail, permettant de prendre des mesures défensives rapides et d'empêcher certaines attaques.

Contents

List of Publications

- **S. Fernandez**, M. Amoretti, F. Restori, M. Korczynski, and A. Duda, "Semantic Identifiers and DNS Names for IoT" in 2021 International Conference on Computer Communications and Networks (ICCCN), (Athens, Greece), pp. 1–9, IEEE, July 2021. [1]
- **S. Fernandez**, M. Korczyński, and A. Duda, "Early Detection of Spam Domains with Passive DNS and SPF," in Passive and Active Measurement, vol. 13210, pp. 30–49, Cham: Springer International Publishing, 2022. [2]
- J. Bayer, Y. Nosyk, O. Hureau, **S. Fernandez**, I. Paulovics, A. Duda, and M. Korczyński, "Study on Domain Name System (DNS) Abuse Appendix 1 – Technical Report," Computing Research Repository (CoRR), 2022. [3]
- Y. Nosyk, O. Hureau, **S. Fernandez**, A. Duda, and M. Korczyński, "Unveiling the Weak Links: Exploring DNS Infrastructure Vulnerabilities and Fortifying Defenses," in 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), (Delft, Netherlands), pp. 546–557, IEEE, July 2023. [4]
- **S. Fernandez**, M. Korczyński, and A. Duda, "WHOIS Right? An Analysis of WHOIS and RDAP Consistency", to be published.
- **S. Fernandez**, Y. Nosyk and M. Korczyński, "Unilateral Opportunistic Deployment of Encrypted Recursive-to-Authoritative DNS", to be published.

Introduction

0.1 Context

The majority of the modern Internet still uses protocols initially designed in the early 1970s, when the Internet was called ARPANET and only connected a few dozen devices across US research institutes. In this early network, just like in today's Internet, devices have to know their destination's Internet Protocol (IP) addresses to send them data. But those addresses were only designed to allow computers to communicate; rather than made to be easily remembered by humans. As a consequence, network administrators of each node in the ARPANET maintained a host file, associating meaningful human-readable names with machine-readable addresses. At the time, the low number of devices connected to the network allowed a team at the Network Information Center (NIC), under the supervision of Elizabeth Feinler, to manually process the declarations of all new devices and provide a unique host file that could be downloaded on each device [5]. However,with the growth and evolution of the network over the years, the limits of a local, centralized, manually -maintained host file arose. This led computer scientists to design a distributed system that could translate domain names to network addresses, the Domain Name System (DNS) [6, 7].

The DNS is a distributed hierarchical naming system, associating domain names with various information. Its main usage is associating domain names to network IP addresses, but it can provide many other kinds of information, among which mail servers, aliases and security parameters. In their most basic form, domain names are a list of text labels, separated by dots [6, 7]. Adding a new label at the beginning of a domain name creates a subdomain, under the authority of the initial domain, for example, the `www.wikipedia.org` subdomain is managed by the `wikipedia.org` domain, itself managed by the `org` Top Level Domain (TLD). Based on this syntax, the DNS is built as a tree data structure where each node has authority on the nodes under it, called its *zone*. The root of this tree defines the set of Top Level Domains (`.net`, `.org`, `.fr`, ...) and delegates the management of these zones to *Registries*, that can be companies (like `.airbus`), organizations (like `.org`) or countries (like `.fr`). Those *Registries* are in charge of adding and removing domains inside their zone and delegating the management of subdomains to the entities that register them, for example, the *Registry* in charge of the `.fr` TLD delegated the `wikipedia.fr` zone to the Wikimedia France foundation.

Registering a new domain name is a process involving multiple entities. We will take the example of a user willing to register the domain `example.fr` to illustrate the different steps. First, the user (called the *Registrant*) contacts a *Registrar* and checks if it is accredited to sell the `example.fr` domain. When the transaction is complete, the *Registrar* contacts the *Registry* in charge of the `.fr` Top Level Domain (TLD) (here, the French Association for Cooperative Internet Naming, AFNIC). The *Registry* adds this new domain to its zone and updates the different DNS entries for this domain, effectively delegating the management of the `example.fr` zone to the *Registrant*. From this point, the domain becomes active, and users that query the DNS for `example.fr` will receive the information linked to this domain. Moreover, the *Registrar* and *Registry* also add registration information in two other places: the Whois and RDAP databases [8, 9]. Since the deployment of the European General Data Protection Regulation (GDPR) [10], *Registrant* personal information may not be provided when the domain is registered by an individual.

The main mechanism that allows the DNS to be distributed is the delegation of authority, with the entity managing a zone delegating parts it to other entities. At the infrastructure level, this delegation is represented as pointers to other servers. To find the IP address of `www.wikipedia.fr`, a user first asks the root of the DNS. The servers that are authoritative for the root zone answer that they delegated the `.fr` zone to AFNIC, and provide the name and IP address of AFNIC's authoritative server. The client then recursively queries AFNIC's server, asking for `www.wikipedia.fr`. This server that is authoritative for the `.fr` zone answers that they delegated the `wikipedia.fr` zone to Wikimedia France, and provides the address of their authoritative server. Finally, the user queries the server authoritative for `wikipedia.fr`, asking for the IP of `www.wikipedia.fr`, and gets the final answer from this server. This chain of delegation of authority allows the domains information to be spread across multiple servers.

This decentralized architecture comes with additional costs. In order to get an information about `www.wikipedia.fr`, a client has to send at least 3 queries, following delegations to 3 different servers, making this system vulnerable to lossy or slow networks. To circumvent this limitation, the Internet Engineering Task Force (IETF), the structure in charge of writing the RFCs defining the different internet protocols, defined a new kind of entity in the DNS infrastructure, the recursive resolvers. Instead of each user having to manually follow all the intermediate delegations from the root to the target domain, they can send their query to a recursive resolver. The resolver will follow the chain of delegation of authority until it reaches the server authoritative for the target domain, and will only return the final answer to the client. This reduces the load on end-users that may have a limited network connection. Additionally, the resolver may cache results in the delegation chain. For example, next time a user asks for a domain under the `.fr` zone, the resolver will already know the address of its authoritative server, and will avoid unnecessary queries to the root servers, thus greatly reducing the number of queries needed and the total time needed to give an answer to the user. Moreover, as multiple users can use the same resolver (e.g. Internet Service Providers can provide default resolvers to their clients), the information gathered to answer one query can be sent to another user asking for the same domain, allowing the resolver to answer a DNS query without having to send any additional requests to authoritative servers.

Between the early 1980s, when the DNS protocol was designed, and today, the Internet evolved rapidly and assumptions that were reasonable at the time no longer hold. The basic versions of DNS has no privacy or security mechanisms: the queries and answers are sent unencrypted and unsigned on the network [6]. This allows malicious actors to see, intercept, and modify both queries and answers: they could learn who queried what domain and are able to impersonate any server, as the users have no way to hide their queries or authenticate the answers. As a consequence, the DNS specifications were later extended to tackle these new network paradigms. To make sure that the answer to a query was not forged by intermediate actors, the IETF defined the Domain Name System Security Extensions (DNSSEC) [11, 12], a way for authoritative servers to sign their answers using public key cryptography and certificate chains to authenticate themselves. To avoid eavesdropper on the network, the IETF designed alternate ways to send queries to a resolver or an authoritative server, adding a layer of cryptography to hide the query and the answers from potentially malicious users. These new protocols are called DNS over TLS (DoT) [13], DNS over HTTPS (DoH) [14] and DNS over QUIC (DoQ) [15], depending on which encryption layer and protocol is used. Those extensions to the DNS protocol are not mandatory, to preserve backward compatibility, but they allow users, resolvers and authoritative servers to improve the integrity, authenticity and privacy of their communications.

Many different systems rely on the domain name architecture and the DNS infrastructure for different purposes. One of them is the mail delivery system. When sending a mail to a target address, the sender must first know the IP address of the mail server in charge of the target domain. This information is stored inside the DNS, and authoritative servers provide the name of the mail server for this domain. As a consequence, the delivery of mails depends on the ability of the DNS system to point users to the right mail server. Following on email delivery, when a user receives an email, they may not be certain that the `From` field of the mail contains the real source of the mail: anyone could have contacted their mail server and forged the sender field as the Simple Mail Transport Protocol (SMTP) [16] provides no way to verify the origin of the email. However, additional protocols using the DNS were defined to allow the receiver to check the identity of the sender, like the Sender Policy Framework (SPF) [17], Domain Keys Identified Mail (DKIM) [18] and Domain -based Message Authentication, Reporting and Conformance (DMARC) [19]. All of these three protocols improve the security of mail systems by allowing the receiving mail server to verify the sender information. This is done by querying the DNS for security details of the alleged sender, like a list of authorised senders or public keys used by the domain to sign all its mails and verify that the origin of the mail is indeed the one claimed in the `From` field of the mail.

The DNS system, as a crucial part of the Internet infrastructure, is often directly the target of cyberattacks, or used as a way to support and deploy malicious activities. Because it is a cornerstone of the Internet, attacks on the DNS can have huge consequences for users, ranging from slow navigation if the servers are the target of Denial of Service (DoS) attacks to complete unavailability of most services if servers are taken down, or spread of malware if servers are compromised. The different security and privacy protocols described previously were designed in response to the increasing number of attacks on the DNS infrastructure and were built to address the limitations of the early implementations. However, as the complexity and global usage of the protocol grows, so does the attack surface and the room for human errors and misconfigurations.

Overall, the DNS is a target of choice for research as it can provide crucial insights on multiple aspects of the Internet. Some studies focus on the technical implementations and performances of the system. Others use the DNS as a way to study the Internet and the protocols associated. However, many research works on the DNS need, at some point, to analyze its content and the traffic between clients and servers. The two main challenges when studying the DNS are the volume of traffic that it represents, and its decentralized structure. At the time of writing, there are almost 400 million unique second-level domains [20], under 1589 different TLDs [21]. Each of the 13 servers authoritative for the root of the DNS architecture receive around 2.5 billion requests per day, for a total query count of more than 30 billion requests per day at the root of DNS [22]. This volume of queries, distributed over thousands of servers all around the world is impossible to fully monitor (let alone process or analyze). As a consequence, studying the DNS is done through indirect means, or sampled data. One way to get a glimpse at the DNS traffic is to monitor parts of the network and log observed queries and answers, and is called *Passive DNS* as we only listen for queries and answers sent by other entities, without interfering with the process or sending anything. For example, this can be done at the boundaries of a company or university network, or at the entrance of recursive resolvers or authoritative servers. These points of measurements provide an almost real-time sample of the DNS traffic, allowing us to observe real events and behaviors like spam campaigns, malware distribution or device misconfiguration, but the data is incomplete and biased (the Internet behavior of a university campus is not representative of the global traffic) and can represent a non-negligible volume of data to store, process and analyze. Another way to study the content of the DNS is to actively scan and probe servers, sending them requests to see how they behave and the kind of data they host. Some authoritative servers even provide open access to their full zone, listing all domains under their authority. These scans and zonefiles are easier to run and analyze, but they do not give information on the actual traffic received by the different servers, they only give information on the content of the servers. Therefore, we used both approaches in order to have a comprehensive understanding of the DNS traffic and the content of the authoritative servers.

For most users, domain names are a way to be visible and present on the Internet, to advertise a brand, a company or an event, or to contact and communicate with people. This is especially the case for companies whose presence is mainly on the Internet, like social networks, search engines, cloud and mail providers, blogs and video sharing platforms. For those services, their domain name *is* their name and identity, it is the way they are known and reached by users. Today, the DNS protocol is a cornerstone of the Internet and resolves billions of domain names per day. As the protocol evolved during the years, it ended up doing much more than just mapping human-readable names to IP addresses. At the time of writing, the DNS is defined by multiple Requests for Comments (RFC) and categorized into 26 different standards, 13 proposed security standards, 13 informational documents and 5 Best Current Practices. All these documents and the previous versions they outdated define the technical inner-workings of the protocol. They describe how all the parties involved in its distributed architecture should work together to provide a fast, resilient, secure and trustworthy system.

To sum up, the Domain Name System is a central piece for the modern Internet and multiple other protocols and systems rely on its structure and availability to support parts of their design, ranging from service discovery to mail security and malicious botnet deployment. Studying the DNS raises many challenges, like the volume of data to analyze, the complex interaction of multiple protocols and systems and the difference between the theoretical protocol and the actual usage of this system. However, it provides unique insights into the behavior of entities on the Internet, the deployment of technologies and protocols and can be used to detect and prevent malicious behaviors.

0.2 Motivations and Contributions

The DNS can be a massive source of information, both for the clients that use it as a distributed database, and for the researchers that use it a tool to study the modern Internet. In this thesis, we explore the DNS as a way to establish trust between entities on the Internet and as a source of information providing valuable insights into the Internet usages and diversity. We explored those aspects and made the following three main contributions:

- We dived into the domain name registration process and challenged the assumptions that the data stored in the DNS, Whois and RDAP databases are always coherent. We gathered millions of DNS, Whois and RDAP entries and compared them to check if their content could be trusted, or if the many systems and research works that rely on this assumption should take extra care when handling this kind of data. We found that while the majority of the data is coherent between those different sources, some fields can be prone to errors, misconfigurations and mismatches. As a consequence, entities that rely on large-scale analysis over millions of domains can use those different data sources interchangeably and expect a low mismatch rate, but entities that need to trust the data of individual domains should take extra care and cross-check the different data sources as they can be incoherent and some fields are more prone to errors.

- We explored how devices in constrained environments, like IoT devices, can use the DNS and domain names to provide complex and efficient discovery systems. We designed a way to name devices that encodes its properties, like the kind of data it provides, or its geographical location, and described how this name can be stored in the DNS in a way that allows easy discovery. With this naming system, users could query the DNS for complex requests, like finding all sensors of a given type in a given area, and get results without putting additional load on the constrained end-devices.

- We studied how domains that are sources of spam mails are configured on the DNS . By analysing their SPF configuration, a protocol improving the security of mail delivery and source verification, we detected that benign domains and spam domains often have different configurations. These security configurations are stored in the DNS, so they can be actively gathered, or observed in real traffic. We built a spam detection system based on these differences between spam and benign domains, leveraging the fact that the DNS entries can be gathered and analyzed before the start of a spam campaign from this domain, allowing to take defensive measures before any victim is targeted.

This manuscript is structured as follows. In Chapter 1, we will describe the DNS infrastructure, its technical design and practical implementations, as well as other systems supported by the DNS. We will present how previous research used the DNS as a tool to observe the Internet. In the following Chapters, we will detail the three main contributions of this work. Each part will have its related background, previous works and conclusion. Then, we will conclude this work and present perspectives for future research on this topic.

1
The Domain Name System

The Domain Name System (DNS) is at the core of the modern Internet, and many protocols and systems rely on its properties to support their own architectures. In order to study how the DNS can be used as a source of trust and information, we first need to understand the basic concepts of this system, how it is deployed and how it is used by many different actors. In this chapter, we will provide the basic information about the DNS and its architecture needed to understand the following chapters that will focus on specific use cases and situations. The structure of this chapter is as follows. First we describe the generic concepts of the DNS, what it is built for. Then we focus on how these concepts are implemented and describe the resulting system architecture. We follow with a few case studies of systems supported by the DNS, like IP resolution and parts of the mail system. The next section describes how malicious actors directly attacked the DNS or used it to support their malicious activities, and how the community improved the protocol to face those threats. Finally, we focus on the different techniques used to observe, monitor and study the DNS and its multiple usages.

Contents

1.1 Generic Concepts

The Domain Name System is a set of multiple interconnected components that interact together to associate information with domain names. Its behavior, protocol and architecture are defined in Request For Comment (RFC), technical documents that define standards and protocols that can be used on the Internet. The RFC publication is managed by the Internet Engineering Task Force (IETF), but each document can be authored by individuals or groups of engineers, computer scientists and researchers. The protocol, architecture and mechanisms of the DNS are defined in multiple RFC documents, like RFC 1034 [6] and RFC 1035 [7]. These documents define the DNS as a *hierarchical and distributed database* mapping domain names to information.

1.1.1 Database

The main role of the DNS is to map domain names to pieces of information. A domain name is technically defined as a set of labels (containing letters, numbers and hyphens), separated by dots, and is used to identify a resource on the Internet, like servers, networks or services [6]. For example, `www.wikipedia.fr` is a domain name made of three labels, and describes the server hosting the web page of the French Wikipedia.

When queried for a given domain name, the DNS provides technical information about this domain in the form of Resource Records (RR). Each record is one entry in the DNS database. There are multiple types of records, depending on the kind of information they hold, and clients can query the DNS for specific record types.

At the time of writing, there are around 50 different types of records, defined over multiple RFCs, each holding a different kind of data. Some of the possible types are:

`A` : the IPv4 address of the domain name

`AAAA` : the IPv6 address of the domain name

`MX` : the name of the servers managing the mails for this domain

`TXT` : a text field for any information or configuration

`DNSKEY` **and** `NSEC3` : holding cryptographic keys and signatures

For example, if a client wants to visit `www.wikipedia.fr`, their computer will not be able to directly connect to the server hosting this website, because it needs its IP address. As a consequence, their computer must first query the DNS for the `A` record of `www.wikipedia.fr`, getting the IPv4 address of the server, and then their browser will open a Transmission Control Protocol (TCP) connection to this IP and download the website pages with the HyperText Transfer Protocol (HTTP).

1.1.2 Hierarchical

As described previously, domain names are made of several labels, separated by dots. The DNS defines hierarchical relations between domains, based on their labels: if a domain is a *suffix* of another, they are called *parent* and *subdomain*, and the parent domain is placed higher in the DNS hierarchy. For example, `www.wikipedia.fr` is a subdomain of `wikipedia.fr`, itself being a subdomain of `fr`. The top of this hierarchy is called the root domain and is represented with the empty string. Domains directly under the root, like `com`, `org` and `fr` are called Top Level Domains (TLD), and domains under these TLDs, like `a.com`, `b.org` and `c.fr` are called Second Level Domains. Figure 1.1 represents the resulting tree architecture.

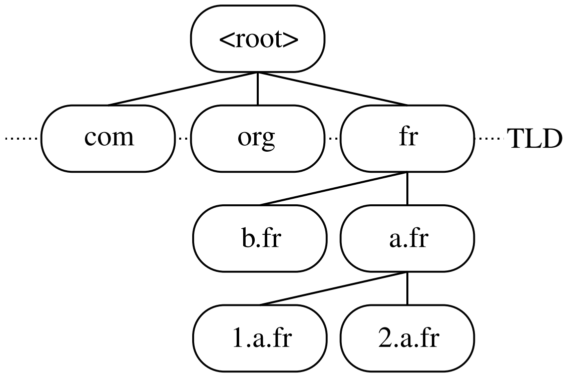


Figure 1.1: Tree structure of the DNS

In the DNS structure, the entity owning a domain has authority over all its subdomains, meaning that it can add, delete or modify any resource records for this domain and all its subdomains. As a consequence, in the structure described in Figure 1.1, the administrators of the `fr` TLD have authority over `a.fr` and `b.fr` but not over `a.org` as it is not in the subtree covered by `fr`.

The entity that has authority over a domain and its subdomains (called a zone) can then delegate this authority to different actors. The root of the DNS is managed by the Internet Assigned Numbers Authority (IANA) but it delegates the authority over the different TLDs to countries, associations and companies. For example, IANA delegated the authority over the `fr` zone to the AFNIC association [23]. Then, AFNIC delegated the authority over `wikipedia.fr` to the French Wikimedia foundation.

In order to register a new domain, a user must contact the entity that has authority over the zone (more specifically, one of their accredited sellers) so it adds a new delegation to the user for this domain. Once this is done, the user has full authority over their zone and can freely add the different resource records for their domain or subdomains.

We describe the registration process and the different entities involved in greater details in Chapter 2 where we focus on how registration information is stored on different databases and can provide crucial insights on the properties of a domain.

1.1.3 Distributed

The Domain Name System is distributed over many different servers all around the world. Each zone can be hosted by multiple servers, called replicas, synchronized to provide the same data. The presence of replicas allow to better distribute the load on multiple servers, avoid single points-of-failure and reduce the latency for users by deploying a replica of the server closer to them.

The authority delegation process also allows the presence of multiple servers answering DNS queries. The entity in charge of a zone can create an `NS` resource record for a domain, pointing to the name server that has authority over this domain. For example, the servers authoritative for the `fr` zone provide the following resource record:

```
wikipedia.fr.  IN NS ns-5-a.gandi.net
```

This entry tells the user that queries to the `wikipedia.fr` zone should be sent to a different server, named `ns-5-a.gandi.net`. As a consequence, the content of the DNS database is not stored in a single server, it is replicated over multiple servers and each zone can delegate the management of sub-zones, redirecting the clients to a different server.

1.2 Architecture

The distributed system described in the previous section involves many different actors and servers in close interaction to provide the resource records queried by the user. The two main types of servers in the DNS architecture are called *Authoritative servers* and *Recursive resolvers* [6].

Authoritative servers are the endpoints holding the resource records for a zone. They are the servers that provide authoritative answers to requests. Each authoritative server has authority over one or several zones (a domain and all its subdomains) and is in charge of providing all the resource records for domains in this zone. As described in the previous section, an authoritative server can delegate its authority over part of their zone to a different authoritative server by adding an `NS` record for the delegated domain pointing to a different name server. A server that has authority over a zone can freely add, remove or modify any record in this zone. To better balance the query load over a zone, there can be multiple authoritative servers for the same zone. This can be done with three main methods: IP anycast, load balancing proxy, and multiple `NS` entries. With IP anycast, all the replicas of the authoritative server share one unique IP address, and queries to this IP are routed to the replica that is the closest to the user in the network, effectively balancing the load over several servers and reducing the ping for the users as the target server is closer to them. User queries can also be sent to a load balancing proxy that will choose one replica of the authoritative server based on the load of each replica and will forward the query. The third main way to replicate the authoritative servers over multiple locations is through the use of multiple `NS` entries: when delegating a zone, the parent zone may provide multiple `NS` entries for the same subdomain, the user should then randomly choose one of them as their target, and can fall back to the other servers if the one they chose is not working properly or query all replicas in parallel to maximize the chances of getting a fast answer [7].

The main drawback of the delegation chain is that in order to resolve a domain clients have to follow multiple redirections, each time sending new packets and waiting for answers, leading to very long resolution times in lossy or slow networks. For example, to resolve `www.wikipedia.fr`, a user first sends a query to the root servers and receives the `NS` entry for the fr zone, they query the `fr` authoritative server and receive the `NS` entry for the `wikipedia.fr` zone, and finally they query the `wikipedia.fr` authoritative server and get the desired answer. This process is represented in Figure 1.2 and involves the exchange of at least 6 packets over the network.

A different kind of server was defined to avoid this load on the clients: *Recursive resolvers*. When queried by a client, these servers will follow the delegation chain until they receive the answer from the authoritative and will only send back the final answer. This method increases the global number of packets exchanged, but reduces the number of packets sent or received by the client. As a consequence, if recursive resolvers have a high-quality network connection to the authoritative servers, the impact of a lossy network on the client side will be greatly reduced. This architecture is described in Figure 1.3. In order to reduce the total number of packets exchanged, each resource record has an attribute called Time To Live (TTL), representing the amount of time (in seconds) during which this entry can be considered as valid and should not

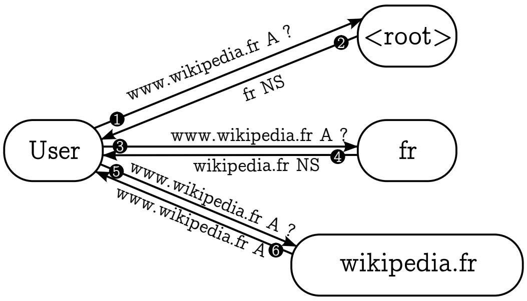


Figure 1.2: Iterative resolution of a domain

be queried again. For example, if the `www.wikipedia.fr A` entry provided by `wikipedia.fr` has a TTL of 5 minutes, the client and the resolver should cache the entry, and for the following 5 minutes, all queries to this entry should be answered from the cache, instead of sending new queries to authoritative servers. `NS` entries also have a TTL value and can be cached. As a consequence, instead of following the whole delegation chain from the root to the queried domain, resolvers and clients can directly use the cached `NS` entry to ask the relevant server. Internet Service Providers (ISP), companies and networks typically provide a recursive resolver to their users, therefore greatly reducing the total number of queries, as resource records from one user can be cached and used to answer queries from another user.

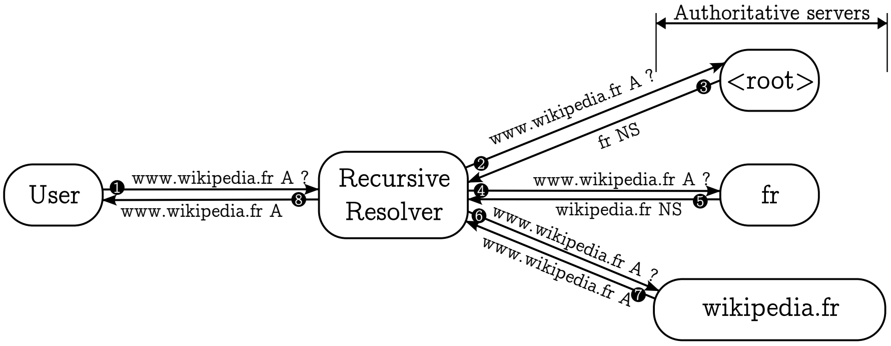


Figure 1.3: Resolution of a domain with a recursive resolver

Recursive resolvers can be *closed* if they are accessible only from inside a given network, like a company, a university or an ISP, and only answer queries from their users, or *open* if they are accessible by anyone on the Internet, and will resolve all queries.

1.3 Example uses of the DNS

In this section we will describe a few classic cases where the DNS is supporting widely used protocols, to highlight its importance in the Internet. First, we will describe its role in IP resolution for several protocols widely used on the Internet. We will then focus on its role in the mail delivery process and how mail servers can use it to increase the security of their systems.

1.3.1 IP Resolution

All protocols in use on the Internet require the devices to know the IP address of their target to send them messages and exchange information. The IP protocol is the basis of all modern protocols, like web browsing with HyperText Transfer Protocol (HTTP(S)) [24], mail delivery with Simple Mail Transfer Protocol (SMTP(S)) [16], remote administration with Secure Shell Protocol (SSH) [25] and file transfers with File Transfer Protocol (FTP) [26]. However, using IP addresses to connect to such services can be tedious as IP addresses are hard to remember and may change over time. As a consequence, most tools for these protocols allow the user to provide a domain name as a target and will use the DNS to get the corresponding IP address to use for the rest of the protocol.

For example, when web browsing, users must provide the URL they want to visit. These strings must have the following format: `<scheme>://<host>/<path>`, with `<scheme>` typically being `http` or `https` for web browsing. When a user wants to visit the home page of the French Wikipedia, they may provide the following URL: `https://www.wikipedia.fr/index.html`, asking their browser to use the HTTPS protocol to connect to `www.wikipedia.fr` and get the `index.html` web page. Their browser will then extract the domain `www.wikipedia.fr` from the URL, query the DNS for the `A` record of this domain and use the resulting IP to establish a TCP connection to the server and send their HTTP packet, asking for the resources stored at `<path>`.

Every time an Internet protocol accepts domain names instead of IP addresses, it will first use the DNS to get the corresponding IP and then use this IP for the rest of the protocol. This has several advantages. First, it allows users to connect to a service without having to remember the complex IP addresses, and instead use human-readable domain names that can be more closely linked to the name of the service they are using (e.g. to visit the Wikipedia page, it is easier to remember `wikipedia.fr` than its IPv4 address). Another advantage is that it allows the IP addresses of servers to change (e.g. when migrating the server to a different host) without the users having to learn the new address: the administrators can just modify the `A` entry of the domain and all users will automatically use the new IP after the expiration of the old value stored in their cache. It also allows administrators to duplicate their servers to multiple IPs by adding multiple `A` records for their domain [7], in which case the users should randomly pick one IP to use, and may fall back to the other IPs if the one they selected was not working, effectively balancing the load over the different replicas and providing fallback servers in case of server maintenance or cyberattacks.

1.3.2 Mail Delivery and Security

When writing a mail message, a user must provide a list of destinations in the form of mail addresses. Mail addresses have the following format: `<local>@<domain>`, representing the mailbox called `<local>`, hosted by the `<domain>` [27]. As a consequence, when delivering the mail, the sender must first determine which server manages the mailboxes of the target domain. Thus, they first query the DNS for the `MX` entry of the `domain`, receiving the name of the mail server. They then send a second query to the DNS, asking for the `A` or `AAAA` entry of this mail server, allowing them to open a TCP connection with this server and transmit the mail using the SMTP protocol. The `MX` entry allows domains to host their mail servers at a different location than their web servers and easily externalize the hosting of this service.

In the basic version of the SMTP protocol, the sender can write any value in the `From` field of the mail. This can be useful for managing multiple aliases or mailing lists, however it also allows anyone to impersonate the sender of a mail. Multiple protocols were designed to avoid this vulnerability, like the Sender Policy Framework (SPF) [17], Domain Keys Identified Mail (DKIM) [18] and Domain-based Message Authentication, Reporting and Conformance (DMARC) [19]. With those protocols, the receiver can verify if the sender really is the one described in the From field of the mail. If the From field of the mail is a@ex.fr, the receiver can query the DNS and get the TXT entry of the ex.fr domain. This resource record can contain any free-form text, but in the case of the SPF, DKIM and DMARC protocols, it holds a list of servers allowed to send mails from this domain, public keys that are used by the sending servers to cryptographically authenticate their mails, and a description of what to do if one of these previous checks are not valid. The receiver can parse this configuration, run the different checks on the sender (its source IP and mail signatures), and determine if they trust the source of this email.

We will go into more details on the SPF protocol in Chapter 4 where we will detect spammers using their SPF configuration.

1.4 Abuses

Its central position in the Internet makes the DNS a target of choice for malicious users. Some attacks specifically target the DNS to disrupt its availability [28, 29] or corrupt its data [30–32] to impact all the users of the protocols described previously. Other malicious actors use the DNS as a way to deploy their attacks or hide their architectures to make it more difficult to take down [33, 34]. In this section we will see three examples of DNS abuses.

1.4.1 DNS Security and Privacy

In its basic version, the DNS protocol uses clear-text queries and answers sent with the User Datagram Protocol (UDP) or Transmission Control Protocol (TCP). Therefore, any entity observing the packets, like the administrators of the wired or Wi-Fi network, can learn what domains are visited by each client and can intercept and modify the queries and answers, allowing them to impersonate websites, redirect mail to their servers and much more [30]. For example, if a user is browsing a website, they first need to resolve the domain name to get its IP address. If a malicious actor intercepts the answer and modifies the IP address to point to a server they manage, the user will connect to the attacker server, without knowing that they are sending packets to the wrong server. The attacker can then deploy a fake login page on this website and get the user credentials or steal authentication tokens. The different steps of this attack are described in Figure 1.4. This attack can be especially dangerous as the URL displayed in the user browser will still be legitimate.

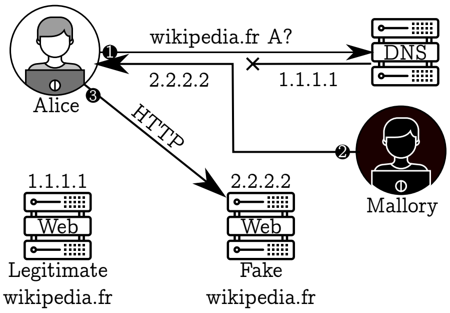


Figure 1.4: DNS answer forging attack

This total absence of security and privacy for the DNS comes from the fact that the base protocol was defined in the very early stages of the Internet, where the threat of cyberattacks was almost inexistant. Then, as more and more systems used the DNS, modifying the protocol to add encryption and digital signatures would introduce breaking changes that were not acceptable by the community. As a consequence, protocol extensions were added to defend against this kind of attacks, but they are optional and not deployed everywhere.

In order to avoid the modification of DNS answers, the DNSSEC protocol extension [12] was defined. This protocol defines several new resource records that allow the managers of a DNS zone to cryptographically sign their records, allowing all users and resolvers to check the integrity of the answers and validate the whole delegation chain, from the root to the final resource record. DNSSEC is already deployed at the root level and by several TLDs [35]. However, it does not provide defense against packet sniffing and requires zone administrators to manage the creation, storage, rollover and retirement of cryptographic keys and can be a source of human errors and a target of different types of attacks.

To prevent actors from reading DNS queries and answers on the network, new ways to connect to recursive resolvers and authoritative servers were designed. Instead of sending clear-text UDP packets to servers, users can add a layer of encryption to hide the query and answer from eavesdropping actors. Three main encryption possibilities are formalized by RFC documents: DNS over HTTPS (DoH) [14], DNS over TLS (DoT) [13] and DNS over QUIC (DoQ) [15]. These protocols encapsulate DNS queries and answers respectively in HTTPS, TLS and QUIC connections, using the encryption layer of these protocols to provide the confidentiality of the communication. These encryption layers are mainly used between users and recursive resolvers, but they are also partially deployed between recursive resolvers and authoritative servers.

We are currently studying the costs and effects of the encryption between resolvers and authoritative servers, but this project is not mature enough to be added to this manuscript.

1.4.2 DDoS Amplification

The main concept behind Denial of Service (DoS) attacks is to overload a target by sending it a high amount of packets and queries [29]. The goal is to exhaust all the target resources (number of connections, bandwidth, memory, CPU time,. . .) to a point where it hinders the queries from legitimate users, or, in the worst cases, completely takes down the target. The main difficulty for DoS attackers is to have enough computing power and bandwidth to send all these queries to overload the target. As a consequence, most DoS attacks are now sent from multiple attack sources, hence the name Distributed DoS (DDoS). One way for attackers to augment their DoS capabilities is to find an amplifying service that will multiply the size of the attack, without consuming too much bandwidth on the attacker side [36]. DNS can be used as an amplification system by forging the source IP of the query packets. The attackers can send DNS queries from spoofed IP addresses, and as a consequence, the answer to this query will be sent to the spoofed IP. However, DNS queries are typically small UDP packets, and DNS answers can be several orders of magnitude bigger as they contain all the queried records. Therefore, an attacker can obtain a high volume traffic by sending only a few forged DNS packets. The basic structure of this attack is described in Figure 1.5. Another advantage of this amplification process is that from the target point of view, the attack comes from the recursive resolver or authoritative server sending them the unsolicited answers, and blocking them could also prevent legitimate users in the targeted network from querying these servers.

A lot of efforts are deployed to avoid this kind of attacks and most servers limit the total answer size or switch to the TCP protocol to send the answer. Contrary to UDP, TCP requires both parties to set up a connection before sending any data. Therefore, the amplification server will try to open a TCP connection with the target to send them the DNS answer, but the target will not accept said connection as it did not send any query to the DNS server. Many additional

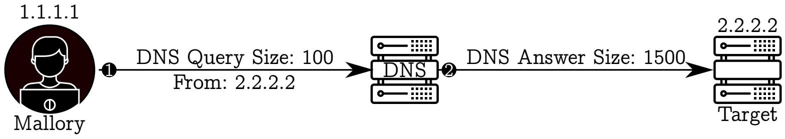


Figure 1.5: DDoS amplification with DNS

amplification defenses are deployed on networks to detect such attacks and reduce their harm, but DNS remains a target for DDoS attackers looking for amplification servers [36].

1.4.3 Botnet Command and Control

Some types of cyberattacks, like DDoS and worms, can involve thousands of different devices on the Internet working together in an automated way, called a botnet. Once they take control of those devices, the main difficulty for attackers is to bring back the data gathered by each device in the botnet and send them commands to execute, especially when the botnet is made of devices infected by a self-propagating virus. Therefore, these infected devices need a way to communicate with a central control point, called Command and Control (C&C) server, to send back data and receive commands. However, if the virus contains the hard-coded IP address of the C&C, security experts can quickly identify the server and take it down, effectively taking down the whole botnet. Even if infected devices instead use domain names to contact the C&C, allowing the attacker to move the C&C to different IPs if one server is taken down, authorities can also take down domains, effectively removing them from the DNS or redirecting their traffic to security analysts to study the botnet behavior. The main counter-measure to this defense mechanism is to generate a high number of domains with a Domain Generation Algorithm (DGA) and embark this algorithm on the infected devices [33]. This way, infected hosts typically use the DGA with the local timestamp to determine which domain to contact for the next communication and use the DNS to resolve this ephemeral domain to the C&C server (or one of its proxies). As a consequence, the domain contacted by infected devices will change over time, making it challenging for authorities to take down each domain. This method is called Fast-Flux domains, and is mainly used to hide botnets from security systems [34].

Most botnet administrators use the DNS to build resilient networks and hide their architecture and C&C servers from security experts. To take down botnets and C&C, security experts can try to take down the C&C server, take down its IP address or the domain names used by infected devices to contact it. However, this can be a difficult task as recent botnets set up a complex domain name architecture, with ever-changing redirections and resource records with small TTLs. The main defense mechanism used today is to get a sample of the virus and reverse-engineer its DGA to predict which domains will be used in the future and take them down before they are used by the botnet. This being said, this method still requires a lot of manual analysis of compiled viruses and is an important topic of research on security [34, 37, 38].

1.5 Observing the DNS

In order to measure how the DNS is used and abused, we need to get reliable sources of information on this system. However, the distributed nature of the DNS architecture makes it challenging to observe or monitor its traffic, as it is spread all over the networks and no single point of measurement can be determined. Each server in the DNS architecture only observes a sample of the total real traffic, therefore, this traffic may not be representative of the worldwide

DNS usage. As a consequence, most works on the DNS rely on partial sources of information and infer global behaviors from the observed samples. The main ways to learn about the DNS content and behavior of the users can be *Passive DNS* and *Active DNS*.

1.5.1 Passive DNS

Passive DNS is a method of observing real DNS traffic in the wild, without sending queries or taking an active part in the protocol. The main sources of *Passive DNS* data are recursive resolvers: they receive queries from users and then recursively query authoritative servers to get the answer and send it back to the user. Some works directly contact recursive resolvers administrators and negotiate access to part of their logs and data, most of the time in the form of traffic captures between the resolver and authoritative servers. This traffic has the advantage of not containing the IP address of the user sending the query, thus greatly limiting the amount of personal data processed during the studies. However, traffic from a single recursive resolver is often heavily biased as it is a single point of observation. For example, measuring traffic from an open public resolver will not capture the traffic from home networks, mainly managed by DNS provided by the different Internet Service Providers (ISP). Moreover, the DNS traffic from a university campus recursive resolver in Europe is inherently different from the traffic observed at a company internal DNS resolvers from China. As a consequence, some platforms like Farsight Security [39] and SIE Europe [40] provide access to an aggregation of multiple traffic sources to reduce this bias and create a feed of DNS messages more representative of the real global traffic. Recursive resolver administrators can send these companies a live duplicated feed of their traffic. The platform then aggregates all the feeds received from their different collaborators and provides access to the resulting aggregated feed where the initial origin of each DNS query can not be traced back. Researchers and experts can then use this feed to observe a live sample of DNS traffic coming from multiple observation points [2,38,41].

The *Passive DNS* aggregation step works as follows. First, each measuring node in companies, universities or open resolvers send a mirror of their raw traffic to the aggregation platform. Then, the first time a domain is observed by the platform, it inserts it in a buffer and announces this insertion. The next time this domain is observed, the aggregator silently increments the domain counter in its buffer. After a fixed amount of time, or when the buffer is full, the aggregator removes the domain from its buffer and announces the domain counter (called `count`) and the timestamps of first and last observations of the domain in the buffer (`time_first` and `time_last` respectively). This aggregation of node traffic into the output feed is illustrated in Figure 1.6.

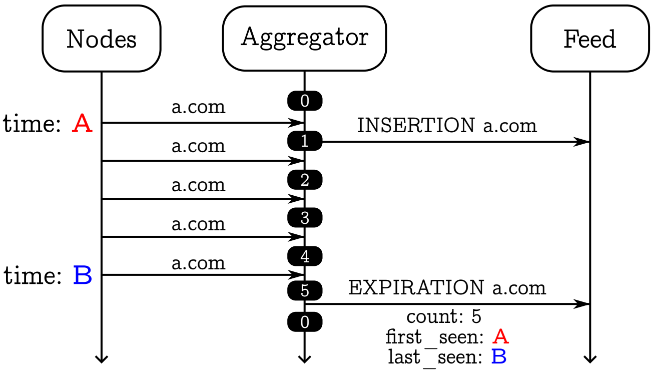


Figure 1.6: Aggregation of node feeds by the Passive DNS platform

This aggregation method greatly reduces the number of packets and total size to analyze: instead of having to receive, store and process thousands of DNS messages, the output feed only sends one message when a domain is first observed, and one message when it leaves the buffer, thus producing a feed that is easier to process. However, due to this aggregation step, we lose some fine-grained information, like the source node of the queries, or the exact time distribution of the queries between `time_first` and `time_last`, reducing the precision of time traffic analysis or query volume evaluations. That being said, *Passive DNS* is a very useful way to observe real DNS traffic, without any interference from the observer, and allows researchers to observe unexpected behaviors in the wild and react quickly when some events are observed in the live-feed. We use this method in Chapter 4, where we monitor *Passive DNS* traffic to gather security configurations of newly registered domains and detect spam domains.

1.5.2 Active DNS

Active DNS is a set of measurements where the researchers or the experts directly send queries to DNS servers. The queries can be addressed to resolvers, authoritative servers or any other entity on the DNS architecture and the sender can then study the answers from these servers. This allows the sender to precisely choose the query to send and ask for specific domains, with specific query parameters, instead of only observing queries sent by other users on the network.

This method is mainly used when the target of the study is a specific server or set of domains because the sender can quickly scan for specific records for the domains of interest, without having to wait for the data to appear in the live feed of Passive DNS. For example, if the study needs the `MX` records of a set of domains, it is more efficient to actively send `MX` queries to their authoritative servers. Some zones also provide direct access to the full list of resource records they hold, called zone file, through different mechanisms like the Centralized Zone Data Service (CZDS) [42] and the AXFR zone transfer [43]. This is the case for the `.com`, `.org`, `.net`, `.ch`, `.li` Top Level Domains: they provide direct access to the list of all their `NS` entries, effectively listing all domains hosted under their TLD. This allows experts to efficiently dive into the content of a zone, instead of having to manually query every single domain to get its content. However, not all zones provide such access methods and active enumeration of domains is often the only way to gather data on specific domains.

The main drawback of *Active DNS* is that it only brings information on the content of the DNS resource records but gives no insights on how or how often these records are queried. However, active scans can be way faster than passive observation of traffic to gather specific data. For example, querying all ~400M domain names for their `A` record can be done in around a week of active scans, allowing for large-scale analysis that would be impossible with Passive DNS [44].

We use *Active DNS* scans in Chapter 2 to gather `NS` entries for a specific set of domains to compare the resulting values to other sources of domain name information. We also use zone transfers in Chapter 4 to download zone files regularly and detect the apparition of newly registered domains and focus our attention on those new domains that are more likely to become spam domains.

1.5.3 Ethics

As presented in Section 1.5.1, *Passive DNS* feeds are aggregated from traffic between recursive resolvers and authoritative servers, therefore, the IP address of the user at the origin of the query is not visible in those feeds. Moreover, Passive DNS aggregators do not publish the locations of their nodes and the aggregation process removes all information about which node observed a given query. Therefore, it is likely impossible to trace back the origin of a given query. In the case of active DNS scans and zone file collection we only query for publicly available data served by recursive resolvers, authoritative servers and their managing entities.

Some previous works highlighted the potential presence of personal data in domain names, originating from misconfigurations or cyberattacks. For example, some DNS queries should only be resolved by local resolvers inside the network, like the `.local` TLD reserved for link-local names and service discovery, but queries to domains ending in `.local` can still be observed in Passive DNS feeds due to misconfigurations of devices or resolvers. However, the work conducted in this thesis only used valid registered domains, and automatically discarded queries to invalid domains, effectively discarding such invalid queries. Moreover, this work focuses on large-scale analysis of millions of domains at a time, using publicly available data, and we never focus on queries to one single domain or the individuals behind it.

When running active DNS scans, we strictly follow the rate limits required by the targets, to avoid overwhelming servers with our enumerations and data collection [44]. Moreover, all the scans were conducted from IPs in the university network, and a website is hosted on those servers, describing who we are, what kind of experiments we are running and why servers could observe scans originating from these sources. We provide a way for any entity targeted by our scans to contact us and be excluded from our scan targets. For example, this could be used by small authoritative servers getting overloaded by our scans and that would like us to reduce our scan speed or completely avoid their servers.

1.6 Conclusion

The Domain Name System is a hierarchical distributed database, mapping multiple pieces of information to domain names. It is used by most protocols that make the modern Internet to build more efficient and resilient architectures. The DNS architecture is distributed over multiple authoritative servers and recursive resolvers to provide resource records to users when queried for domains. However, malicious actors also rely on the DNS to hide their activities and propagate their attacks, or directly attack the DNS to take down this crucial piece of the Internet or infect it with malicious content. The DNS is the target of many research studies: new ways to use it, how performant it is, how attackers abuse it or how it can be defended against these attacks. Using publicly available data and passive observations of the traffic, we study several different aspects of the DNS and its place in the modern Internet.

2

WHOIS Right? An Analysis of WHOIS and RDAP Consistency

Public registration information on domain names, such as the accredited registrar, the domain name expiration date, or the abuse contact is crucial for many security tasks, from automated abuse notifications to botnet or phishing detection and classification systems. Various domain registration data is usually accessible through the WHOIS or RDAP protocols—a priori they provide the same data but use distinct formats and communication protocols. While WHOIS aims to provide human-readable data, RDAP uses a machine-readable format. Therefore, deciding which protocol to use is generally considered a straightforward technical choice, depending on the use case and the required automation and security level. In this paper, we examine the core assumption that WHOIS and RDAP offer the same data and that users can query them interchangeably. By collecting, processing, and comparing 164 million entries for a sample of 55 million domain names, we reveal that while the data obtained through WHOIS and RDAP is generally consistent, 7.6% of the observed domains still present inconsistent data on critical fields like nameservers, IANA ID, or creation date. Such inconsistency should be carefully considered by the security actors that rely on the accuracy of these fields.

This chapter is based on a work with Olivier Hureau, Maciej Korczynski and Andrzej Duda and is not published yet.

Contents

2.1 Introduction

Malicious activities such as phishing scams, botnet operations, or malware distribution often involve the use of domain names. To investigate these activities and mitigate their impact, it is crucial to have access to specific information about domain registration. Essential information for investigating malicious activities related to domain names encompasses details such as the domain creation date, the registrant name, the sponsoring registrar, the domain status, the expiration date, email addresses designated for reporting domain name abuse, and other relevant data. However, in compliance with the European General Data Protection Regulation (GDPR) [10] and the Temporary Specification of the Internet Corporation for Assigned Names and Numbers (ICANN) for generic Top-Level Domain (gTLD) registration data [45], personal information pertaining to registrants is typically obscured or hidden.

Different entities involved in the domain registration process typically provide registration information through two protocols: WHOIS [8] and RDAP (Registration Data Access Protocol) [46]. Despite the historical reasons for the co-existence of two protocols, each having its own specific format, and theoretically providing access to the same data, numerous studies [47–50] raised valid concern about the effectiveness and drawbacks of both protocols.

While both protocols were designed to provide registration information, there are no formal requirements mandating consistent results across different data sources. In practice, the registration data may vary between TLD registries, and registrars, as well as between the responses obtained from WHOIS and RDAP. This variability introduces an element of unpredictability with respect to the consistency and accuracy of the provided information. Furthermore, studies that use registration data tend to favor one protocol over the other without providing explicit justification, and they base their preference on factors such as data retrieval speed, parsing capabilities, the presence of WHOIS and RDAP records for each domain, and other convenience-related considerations. Hence, an important issue emerges: to what degree do both protocols offer consistent information? Addressing this question requires a thorough and comprehensive analysis of how the data provided by the WHOIS and RDAP protocols align with each other.

To our knowledge, no previous research examined the assumption that information provided by WHOIS and RDAP is consistent. Nevertheless, many articles put forth classification algorithms, conducted studies on the domain behavior, or initiated abuse and vulnerability notification campaigns relying on data obtained through these protocols. In doing so, they implicitly depend on the accuracy and consistency of the information provided by WHOIS and RDAP.

Our paper makes the following contributions:

- We provide an overview of the disparities between WHOIS and RDAP, shedding light on the rationale behind the coexistence of multiple servers and protocols for accessing registration data. Delving into the historical and technical aspects, we highlight the intricate choices that have led to the current state of uncertainty surrounding the assurance of data consistency.

- We undertake a comprehensive data collection encompassing WHOIS and RDAP records for more than 55 million domains. Our focus is on parsing the fields commonly used in security and privacy studies. We will contribute all the collected registration data to the research community. We perform a thorough analysis of the parsed fields evaluating their consistency and deliberating over potential factors contributing to content variations. By doing so, we aim to raise awareness within the community about the importance of exercising caution with trust in registration data as 7.6% of the observed domains presented inconsistencies in fields used by security and privacy studies.

- We conduct a comprehensive analysis of the nameservers field, cross-referencing the gathered data with the results obtained from active DNS measurements. Our aim is to determine which data source, whether WHOIS or RDAP, is more likely to provide accurate and trustful information.

2.2 Background

We begin by providing background information on the administration of domain names and the collaborative processes within the DNS ecosystem. Delving into the history of WHOIS and RDAP, we explore the reasons for their coexistence. Furthermore, we explain how to access registration data through both protocols, providing a clear outline of their respective procedures. Lastly, we elaborate on diverse approaches and challenges related to parsing WHOIS and RDAP.

2.2.1 The Ecosystem of Domain Management and Registration

The administration of a domain name entails the collaboration of multiple actors who collectively ensure the provision of all the necessary technical and administrative records vital for its operational use. At the top of the Domain Name System (DNS), the Internet Assigned Numbers Authority (IANA) manages the root nameservers and delegates the management of each top-level domain (TLD) to different registries. Country-code top-level domains (ccTLDs) such as `.uk` and `.fr` are managed by country-specific organizations (registries) like Nominet (for `.uk`) or AFNIC (for `.fr`). In contrast, generic top-level domains (gTLDs) such as `.com` and `.business` can be managed by any organization that meets the necessary requirements [51] and obtains authorization from the Internet Corporation for Assigned Names and Numbers (ICANN), like VeriSign Inc. (for `.com`) or Identity Digital (for `.business`). Registries are responsible for managing their top-level domain zones and have the authority to create new domains under their TLD. Each registry delegates the task of registering new domains to registrars, responsible for selling domains to users, referred to as registrants. When contacted by users, registrars collect and centralize user information, and communicate with the registry. In the interaction between registrars and registries, a variety of protocols may be used with the Extensible Provisioning Protocol (EPP) [52] commonly used for seamless communication. The registry then generates the required records such as DNS ones and administrative details to create the domain. For gTLDs under the ICANN agreement [51] and the majority of ccTLDs, both the registry and the registrar make the registration information available to the public. This information is typically accessible through the WHOIS and/or RDAP protocols.

2.2.2 Why Two Different Systems?

The existing WHOIS protocol as defined in RFC 3912 [8] published in 2004 formalized a practice in use since 1982 [53]. RFC 3912 established the guidelines on how a server could offer the information about various Internet entities, including users, servers, domains, and IP addresses with a straightforward query/response protocol. However, it recognized that the WHOIS protocol had certain deficiencies in terms of crucial design goals like internationalization and robust security, typically expected of IETF protocols. RFC 3912 explicitly stated that it did not address these shortcomings and only required the content to be presented in a human-readable format. The decision to retain the original design flaws in the WHOIS protocol can be attributed to historical reasons. The original WHOIS system in use since the early 80s was already implemented on numerous servers. To maintain backward compatibility and prevent disruption to existing systems and practices, the IETF chose to accept the original design flaws rather than mandating widespread changes. This approach aimed to mitigate the risk of a new protocol facing low adoption rates, similar to what occurred with the `SPF` DNS record [17].

After several years, the IETF initiated efforts to design a new protocol aimed at providing domain registration information while addressing the limitations of WHOIS. This endeavor culminated in 2015 in the publication of RFC 7482 [54] that specified RDAP. RFC 7482 [54], along with subsequent extensions [9, 46, 55–57], specifies the protocol emphasizing the provision of machine-readable data in the JSON format. It defines data types, keys, and encoding to ensure structured information. Despite the introduction of RDAP, the WHOIS protocol has not been replaced, and both protocols continue to coexist, offering comparable data.

2.2.3 Data Access and Availability

RFC 3912 [8] and RFC 8521 [58] define the WHOIS and RDAP data access protocols, respectively. The RDAP protocol operates over HTTP(s) using the REST paradigm and returns data in JSON format, while a WHOIS user needs to connect to a server over TCP on port 43 and receive a plain text response.

The registration data may be incomplete, and some registries may only offer minimal information, in this case, they are called “thin”, in opposition to “thick” registries that directly provide the full registration data. This difference in the completeness of registration data remains valid for both WHOIS and RDAP. For instance, the `.com` registry provides minimal information and does not include the registrant organization data. To obtain complete information (with respect to GDPR), the user of both protocols may need to follow referrals to one or several servers (see Figure 2.2): they first need to locate the registry server (①), then submit a query to the registry to obtain the registration information (②), and optionally, retrieve more detailed data from the registrar (③).

For WHOIS queries, users can rely on command line tools provided by their system to bundle most steps and referrals, like the Debian *whois* package. On the contrary, there is no widely deployed command line tool to query RDAP databases.

The user needs to follow the steps below to retrieve registration information of `google.com` using RDAP:

① The user begins by retrieving the bootstrap configuration file from IANA,[1] as specified in RFC 9224. From this file, they obtain the URI of the `.com` RDAP server.

② The user appends the string `domain/google.com` to the server URI obtained in step ①, and forms the query to retrieve the registry RDAP answer at `https://rdap.verisign.com/com/v1/domain/google.com`

- (an illustration of the result can be found in Figure 2.1).

③ The returned JSON object contains a referral to the registrar server (in this example, MarkMonitor, Inc). The user can access this information at `https://rdap.markmonitor.com/rdap/domain/google.com`.

[1] `https://data.iana.org/rdap/dns.json`

```
{
 "objectClassName": "domain",
 "ldhName": "GOOGLE.COM",
 "links": [{
     "value": "https://rdap.verisign.com/com/v1/domain/GOOGLE.COM",
     "rel": "self",
     "href": "https://rdap.verisign.com/com/v1/domain/GOOGLE.COM",
     "type": "application/rdap+json"
   },{
     "value": "https://rdap.markmonitor.com/rdap/domain/GOOGLE.COM",
     "rel": "related",
     "href": "https://rdap.markmonitor.com/rdap/domain/GOOGLE.COM",
     "type": "application/rdap+json"}],
 "entities": [{
     "objectClassName": "entity",
     "handle": "292",
     "roles": ["registrar"],
     "publicIds": [{"type": "IANA Registrar ID","identifier": "292"}],
     "vcardArray": [
       "vcard", [
         ["version",{},"text","4.0"],
         ["fn",{},"text","MarkMonitor Inc."]]],
     "entities": [{
         "objectClassName": "entity",
         "roles": ["abuse"],
         "vcardArray": ["vcard",[
             ["version",{},"text","4.0"],
             ["fn",{},"text",""],
             ["tel",{"type": "voice"},"uri","tel:+1.2086851750"],
             ["email",{},"text","abusecomplaints@markmonitor.com"]]]}]}],
 "events": [
   {"eventAction": "registration", "eventDate": "1997-09-15T04:00:00Z"},
   {"eventAction": "expiration", "eventDate": "2028-09-14T04:00:00Z"},
   {"eventAction": "last changed", "eventDate": "2019-09-09T15:39:04Z"},
   {"eventAction": "last update of RDAP database",
       "eventDate": "2023-05-26T13:57:10Z"}],
 "nameservers": [
   {"objectClassName": "nameserver","ldhName": "NS1.GOOGLE.COM"},
   {"objectClassName": "nameserver","ldhName": "NS2.GOOGLE.COM"},
   {"objectClassName": "nameserver","ldhName": "NS3.GOOGLE.COM"},
   {"objectClassName": "nameserver","ldhName": "NS4.GOOGLE.COM"}],
 }
```

Figure 2.1: Part of the Registry RDAP entry of `google.com` collected at VeriSign server

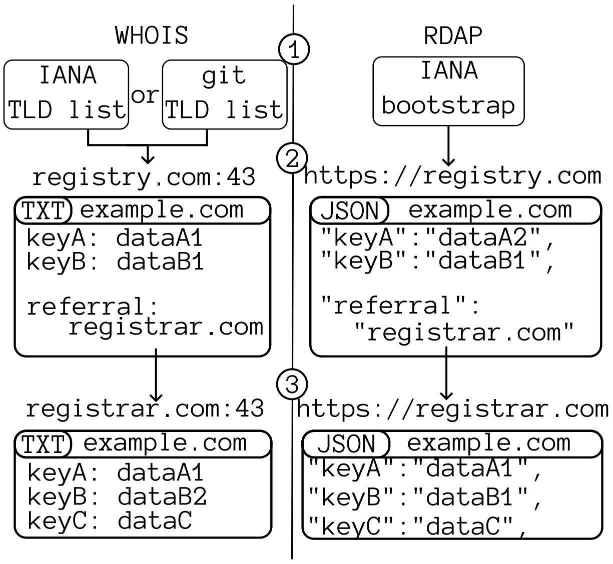


Figure 2.2: Referral system to obtain complete registration data

For WHOIS, RFC 3912 [8] does not provide a bootstrap file for step ①. Instead, users can query the IANA WHOIS server at `whois.iana.org` to retrieve TLD-related information. The response includes the details about the TLD registry, in particular, the domain name of the WHOIS server for that zone. As an example, let us examine the procedure involved in retrieving the registration information for the domain `google.com` using the WHOIS protocol:

① The user proceeds by querying the IANA WHOIS server for the `.com` TLD and locates the record `whois: whois.verisign-grs.com`. This information directs them to the VeriSign server.

② Next, the user queries this server that provides registry WHOIS information for the domain `google.com` (the result is presented in Figure 2.3).

③ Within this record, there is a referral to the registrar server `WHOIS Server: whois.markmonitor.com`. The user can retrieve the most detailed registration data by querying this registrar WHOIS server.

Nevertheless, users may encounter problems when following this approach:

- Certain WHOIS servers may require specific query flags. For example, the WHOIS server for the `.de` TLD expects the flags “`-T dn,ace`”.
- The IANA database may not always be up to date, resulting in inaccurate information about certain TLDs. For example, it does not provide a WHOIS server for the `.cm` TLD.
- In some cases, the TLD registry may not handle the registration information for domain names associated with public suffixes. For instance, the registry server `whois.nic.uk` for the `.uk` TLD does not manage the `.ac.uk` TLD, managed instead by `whois.nic.ac.uk`.

For these reasons, the Debian *whois* package[2] adopts a different approach. It uses a dedicated database that specifies servers responsible for the public suffixes and the corresponding

[2] https://tracker.debian.org/pkg/whois

```
Domain Name: GOOGLE.COM
Registry Domain ID: 2138514_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.markmonitor.com
Registrar URL: http://www.markmonitor.com
Updated Date: 2019-09-09T15:39:04Z
Creation Date: 1997-09-15T04:00:00Z
Registry Expiry Date: 2028-09-14T04:00:00Z
Registrar: MarkMonitor Inc.
Registrar IANA ID: 292
Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
Registrar Abuse Contact Phone: +1.2086851750
Domain Status: clientDeleteProhibited
Domain Status: clientTransferProhibited
Domain Status: clientUpdateProhibited
Domain Status: serverDeleteProhibited
Domain Status: serverTransferProhibited
Domain Status: serverUpdateProhibited
Name Server: NS1.GOOGLE.COM
Name Server: NS2.GOOGLE.COM
Name Server: NS3.GOOGLE.COM
Name Server: NS4.GOOGLE.COM
DNSSEC: unsigned
```

Figure 2.3: Registry WHOIS entry of `google.com` collected at VeriSign server

Table 2.1: Number of active TLDs providing RDAP and WHOIS servers

	RDAP	WHOIS	
Source	Boostrap	IANA	GitHub
ccTLD (309)	27 (9%)	222 (72%)	231 (75%)
gTLD (1152)	1152 (100%)	999 (86%)	1147 (99%)

flags to be used. The source code for this package is accessible in a collaborative GitHub repository.[3] While the repository allows anyone to propose modifications, it has been mainly maintained by Marco d'Itri since 1999. This repository serves as a valuable alternative to the IANA WHOIS server, acting as a reliable starting point for retrieving WHOIS information (referred to as the `git TLD list` in Figure 2.2, step ①).

We have retrieved the information from the RDAP bootstrap file, the GitHub repository of the *whois* package, and queried the server `whois.iana.org` for all active gTLD and ccTLD listed on the IANA website. Table 2.1 shows that the GitHub repository provides 148 additional WHOIS servers compared to the IANA list. For instance, it includes a WHOIS server for the `.cm` TLD, not available on `whois.iana.org`. The table also highlights the proportion of active gTLDs and ccTLDs that offer WHOIS and RDAP services. It is important to highlight that ccTLDs provide relatively less access to registration data than gTLDs. In particular, the adoption of the RDAP protocol among ccTLDs is significantly low, accounting for only 9%. We can attribute the disparity between ccTLDs and gTLDs to the agreement established between gTLDs and ICANN [51]. As per this agreement, registries have to offer access to registration data through the RDAP protocol. However, it does not require gTLDs to maintain WHOIS servers, and it does not apply to ccTLDs. Contrarily, the deployment of RDAP by ccTLD registries is influenced by various factors such as voluntary adoption, local regulations, and technical considerations.

2.2.4 Parsing Registration Data

One of the primary motivations behind the design of RDAP is to address the inherent limitations of the WHOIS system, in particular, its vague and loosely defined "human-readable" format for data. By incorporating the JSON-structured response format and well-defined data element features, among others, RDAP provides a more standardized, machine-readable approach to accessing registration data. This enhancement significantly improves the efficiency and reliability of parsing and extracting information from RDAP responses when compared to the traditional WHOIS system.

WHOIS data has been presented in various formats, undergone frequent changes, and may even be expressed in the local language of the registrar or TLD registry (e.g., the Bolivian ccTLD `.bo` WHOIS records are written in Spanish). The absence of normalization or implicit conventions raises a significant challenge when parsing WHOIS records, as highlighted in the studies that use WHOIS data [47,49,50,59–61].

We can categorize traditional algorithms for parsing WHOIS data into two distinct ap-

[3] https://github.com/rfc1036/whois

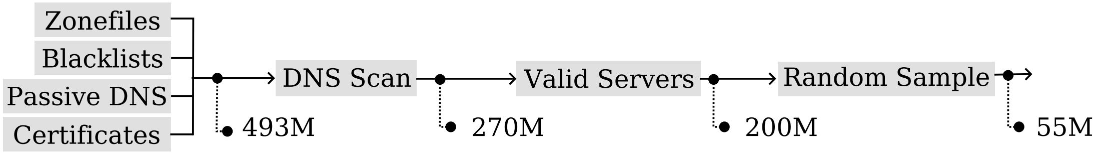


Figure 2.4: The stages of domain selection with the number of domains at each step

proaches: templates and rules. The template-based approaches, such as `Net::Whois`[4] (Perl), `whoisrb`[5] (Ruby), and `PHPWhois`[6] (PHP), offer regular expression templates specifically tailored to each registry or registrar. When using this approach, the user obtains WHOIS data from the registry, parses it using the relevant template for the TLD and registry, extracts any potential referral link to a registrar WHOIS server, and then retrieves and parses the registrar WHOIS data using the corresponding template. This approach is effective when the templates are available and regularly maintained. However, it becomes challenging when no template is available for a specific entity or if the format undergoes changes. Therefore, its success heavily relies on the quantity and quality of the templates, necessitating manual updates for each template.

Rule-based approaches such as `python-whois`[7] use a collection of predefined rules, regular expressions, and Natural Language Processing techniques to identify prevalent formats found in WHOIS records such as `Key: Value`, and extract as many fields as feasible. This approach is versatile and can be applied to any registrar without the need for dedicated templates. It may also accommodate format changes over time. However, it is generally less efficient compared to the use of custom-made templates [61].

Previous work explored existing parsers to train machine-learning algorithms based on Natural Language Processing or used techniques like Conditional Random Field [62] to automatically deduce the data structure and enhance the accuracy of field extraction. This approach demonstrated improved capabilities in extracting various fields from data.

While the template-based and rule-based approaches offer some potential for obtaining registration data through WHOIS, they require regular maintenance and may be less efficient than RDAP. The introduction of RDAP offers a promising alternative for enhanced parsing efficiency and accuracy.

2.3 Methodology

In this section, we outline our methodology for collecting and parsing WHOIS and RDAP records. Considering the significant volume of data, we have meticulously designed our scheme to efficiently collect and parse registration data for a large number of domains within a reasonable time frame. All this is achieved while ensuring that WHOIS and RDAP servers experience minimal strain. We begin by explaining the process of domain selection, as illustrated in Figure 2.4, followed by a comprehensive description of the WHOIS and RDAP parsing process. Lastly, we provide an overview of how we have identified and analyzed discrepancies among the records.

[4] https://metacpan.org/pod/Net::Whois
[5] https://whoisrb.org/
[6] https://github.com/SimpleUpdates/phpwhois
[7] https://pypi.org/project/python-whois/

2.3.1 Domain Data Collection and Filtering

Compilation of registered domain names.

First, we gathered an extensive list of domains by consolidating multiple data sources:

- gTLD zone files obtained from ICANN Centralized Zone Data Service (CZDS)[8],
- ccTLD zone files accessible via AXFR zone transfers (.se, .nu, .li, .ch),
- Passive DNS feed from SIE Europe[9],
- Domain blacklists including SpamHaus[10], APWG[11], OpenPhish[12], URLHaus[13], ThreatFox[14], and SURBL[15],
- Google Certificate Transparency Logs[16], which we continuously monitored to identify newly issued Transport Layer Security (TLS) certificates and extract the corresponding domain names.

All the collected domains are aggregated and deduplicated, resulting in a list of 493 million unique domain names. To guarantee the inclusion of only registered domains, we performed an active DNS scan on each domain, querying for A resource records using zdns [44], and exclude those for which the response is NXDOMAIN (non-existent domain).

Filtering domains with valid WHOIS and RDAP servers.

To study the inconsistencies between WHOIS and RDAP records, we carefully filtered out domains that lacked a recognized WHOIS or RDAP server. This filtering process involved cross-referencing the official IANA list [23] and the GitHub repository, as detailed in Section 2.2.3. After this filtering step, our dataset comprised 200 million domain names.

Scanning all 200M domains would be a time-consuming process spanning several months, along with significant storage challenges. To address this, we opted to work with a representative subset of domains. This subset was randomly chosen from the pool of 200 million domains, with a sample size of 55 million domains carefully determined to facilitate the collection and parsing of WHOIS and RDAP records within a one-month time frame.

2.3.2 Gathering and Parsing Resgistration Data

Data collection.

After identifying WHOIS and RDAP servers for the sampled domain names, we proceeded with the collection of the corresponding records. We gathered the registration data of the selected

[8] https://czds.icann.org
[9] http://sie-europe.net
[10] https://www.spamhaus.org
[11] https://apwg.org
[12] https://openphish.com
[13] https://urlhaus.abuse.ch
[14] https://threatfox.abuse.ch
[15] https://surbl.org
[16] https://certstream.calidog.io

domains between December 6th and December 31st, 2022. During the collection process, we parsed each record to determine if it belonged to a “thin” registry that delegated a part of the data to a referral server, and follow the eventual referral. This step was iteratively repeated to ensure we obtained all versions of the registration data, following all referrals. At the end, we successfully collected a total of 164 million unique records, covering information from over 55 million distinct domains.

To ensure accurate comparisons, we collected WHOIS and RDAP records of each domain within a narrow time window, typically under 1 minute. This prevents the comparison of records collected at different times and reduces discrepancies resulting from domain updates during the scanning process. Moreover, some registrars impose query limits on IP addresses and enforce timeouts or blacklist IP addresses that exceed these limits. To ensure compliance and prevent any disruptions, we adjusted our data collection speed accordingly.

After the collection process, we carefully examined the gathered WHOIS and RDAP records. Any malformed responses (like invalid HTTP packets or JSON objects for RDAP) or timeouts were discarded, while valid responses underwent parsing for further analysis.

Table 2.2: Fields extracted from WHOIS and RDAP records

Field	Data type	Missing rate		Domain inconsistency	Used by
		Records	Domains		
`Nameservers`	Text	3.2%	6.6%	573,790 (1%)	[63–65]
`IANA ID`	Integer	5.9%	13.7%	106,813 (0.2%)	[63, 66–68]
`Creation date`	Date	0.8%	2.2%	3,138,024 (5.7%)	[59, 66–68]
`Expiration date`	Date	1.0%	2.7%	2,424,951 (4.4%)	[47, 67]
`Emails`	Email	7.9%	14.8%	18,958,821 (34.5%)	[49, 50, 59, 60, 63, 67–69]

Parsing WHOIS.

Parsing WHOIS data and extracting all pertinent fields presents a challenge, as detailed in Section 2.2. Consequently, this study focuses on specific fields used in previous research (see Table 2.2), using custom templates designed to accurately parse various formats. We developed 242 custom templates comprising regular expressions that outline the extraction process for selected fields from WHOIS records across numerous registrars. The templates are designed to handle multiple languages and formats, maximizing the comparability of records.

Parsing RDAP.

Contrasted with WHOIS, parsing RDAP records is typically more straightforward, primarily due to the JSON format. Nevertheless, despite the data format being defined in RFC 9083 [46], there might be ambiguity regarding the correct placement of information within the data structure. Consequently, different registries and registrars may have varying interpretations of where specific information should be located.

We gathered the designated fields from all locations allowed by the RFC. We considered malformed fields, those containing incorrect data types, or located in the wrong place within the data structure as missing. For instance, there are two primary representations of domain names in RDAP: as a string object (e.g., `ns.example.com`) or as an array of labels (e.g., `[ns, example, com]`). However, according to RFC 9083 [46], when listing domain nameservers, they must be in the string format. Therefore, if we encountered a nameserver in the array format instead of the expected string format, we considered it as missing. This decision was based on the assumption that most automated systems would adhere to the RFC and disregard the field due to its invalid type.

Field selection.

To compare different data sources, it is important to note that not all registration data records share the same set of fields. As a result, we selected a limited number of fields, which have been commonly used in previous security studies and are consistently present in both WHOIS and RDAP records, whether at the registry or registrar levels. Table 2.2 presents the selected fields, along with the type of data they hold and the articles that have used them. For this research, we have chosen the following fields:

- `Nameservers`: this field indicates the name servers that have the authority over a particular domain.
- `IANA ID` and `Registrar`: the sponsoring registrar responsible for managing the domain is captured in the `Registrar` text field. Additionally, the `IANA ID` is an integer field that typically represents the unique identifier assigned by IANA [70] to each ICANN-accredited registrar (if applicable).
- `Creation date` and `Expiration date`: these fields denote the date of the initial registration for the domain and the subsequent expiration date. Once the registration expires, the domain becomes available for purchase again unless the owner renews it.
- `Emails:` This field contains a range of contact email addresses that can be used, for instance, for reporting domain-related abuse.

We deliberately omitted selecting fields associated with a registrant, despite their use in several studies, due to their absence in many registries. Furthermore, the implementation of the European General Data Protection Regulation (GDPR) resulted in the removal or redaction of the field content by most servers. The impact of GDPR on the content of these fields falls outside the scope of this paper and has already been analyzed in prior research [49].

When a field is absent from a record, or the content could not be parsed, the data is marked as missing. Table 2.2 shows the proportion of records missing each field. The record missing rate indicates the proportion of records with missing data, whereas the domain missing rate represents the percentage of domains that have at least one record with missing data. This considers that each domain has multiple records (i.e., WHOIS and RDAP, including records collected by following referrals).

The missing rates for all fields, except for the IANA ID and Emails fields, are relatively low. This result was expected since the IANA ID solely pertains to domains under generic TLDs and ICANN-accredited registrars. Furthermore, each field presented its own set of parsing challenges, particularly in the case of WHOIS records, but also for RDAP. In RDAP, certain records, such as email contact addresses, can be located in different parts of the JSON structure as defined by RFC 7483 [55].

2.3.3 Analyzing Data Consistency

After collecting, parsing, and cleaning the registration data for all studied domains, we analyzed the consistency among various WHOIS and RDAP records.

For a given domain, if we were able to collect registration data from multiple sources and if these records have common fields, we evaluated the consistency of the data. If the formatted data in same fields is identical, we considered them to be matching fields. On the other hand, if there is a discrepancy between the data, it results in a mismatch. We consider two types of mismatches: the first one involves two records from the same protocol, such as the registry WHOIS not aligning with the registrar WHOIS. The second type involves two records from different protocols, for instance, the registrar WHOIS not corresponding to the registrar RDAP.

2.3.4 Ethical Consideration

We adhered to the best practices recommended by the measurement community to ensure reliable results with minimal disruption to the servers [71, 72]. When gathering various data sources, including WHOIS, RDAP, and DNS records, we meticulously adhered to server rate limits [44]. Additionally, upon visiting the scanner's source IP address, users are presented with a webpage that provides information about our identity, work, and instructions for adding a scanned server to our opt-out lists, allowing them to cease receiving requests from us. Throughout the study, we did not receive any opt-out requests via email.

The raw data we collected may include information about registrants. However, after the implementation of GDPR, most registrars provide options for their customers to choose which fields are visible or automatically redact personal information. In practice, most fields that could potentially contain personal data were redacted by default.

2.4 Results

In this section, we present the analysis of inconsistencies and explore the root causes of the disparities observed in specific fields. Table 2.2 provides a breakdown, field by field, indicating the count of records where the field was missing, the number of domains in which at least one mismatch was identified, or if the field was entirely absent from the records. Excluding the `emails` field, which raises its unique challenges discussed in Section 2.4.3, we observed that 7.6% of all examined domains exhibited at least one inconsistency in the remaining fields.

2.4.1 Nameservers

The typical method to obtain a list of authoritative nameservers for a given domain involves sending recursive queries within the DNS tree, starting from the root zone and progressing toward the registry nameserver, which then provides the relevant information [68]. However, in certain prior studies that had a primary focus on detecting malicious domains [63–65], the nameserver information used in the analysis was obtained from WHOIS.

The primary purpose of the nameserver fields was either to cluster domains with identical nameservers [63, 64] or to conduct further analysis on the nameserver itself. For instance, investigations could involve verifying whether the nameserver is self-hosted, such as `ns.example.com` being authoritative for `example.com`, determining if it is managed by wellkown DNS service operators, or identifying if the apex domain of the nameserver is newly registered [65].

In the subsequent part of this section, we begin by examining the various types of nameserver mismatches and their frequency. Then, we use DNS as a reference point to ascertain the accuracy of the data sources involved in cases of mismatches.

Table 2.3: Number of records and domains with mismatching nameservers

Case	Records	Domains
All	1,044,268	576,204
Inclusion	314,633 (30.1%)	224,833 (39.1%)
Intersection	48,693 (4.6%)	23,934 (4.1%)
Disjoint	680,942 (65.2%)	343,994 (60.0%)

Mismatch Types.

We identified a total of 1,044,268 mismatches between two registration records of the same domain, encompassing 576,204 unique domain names. This accounts for approximately 1% of the overall collected domains; hence 99% of the measured domains did not have mismatching nameservers records.

When the nameservers of two records (referred to as A and B) are found to be inconsistent, three potential scenarios may arise:

Inclusion. $A \subset B$ or $A \supset B$: one set is a subset of the other one.

Intersection. No inclusion but $A \cap B \neq \emptyset$: A and B do not match but they have at least one server in common.

Disjoint. $A \cap B = \emptyset$: A and B have no nameserver in common.

Table 2.3 presents the number of mismatches detected in each scenario. As described in Section 2.3.3, a given domain may have multiple records for each protocol, as each registration record may contain a referral field. As a result, each domain can exhibit multiple types of mismatches. For example, the nameservers extracted from the registrar's WHOIS record could be included in the list of nameservers found in the registrar's RDAP record, and additionally, the nameservers listed in the registry's WHOIS record could entirely differ from the servers in the registry's RDAP record. In such cases, a domain would be counted in both the inclusion and disjoint categories. Consequently, the values in the Domains column may exceed 100%.

When using DNS to fetch a domain's resource records, if the client (e.g., a recursive resolver) has multiple nameservers to choose from, it can use any of them interchangeably or query all and process the first received answer [7]. This means that the inclusion and intersection cases may be less worrisome, as both records share at least one nameserver, potentially indicating that all nameservers serve the same data. Conversely, the disjoint case, in which both records have no servers in common, is concerning as it raises suspicion that the nameservers may not serve the same data or be authoritative for the domain name. This situation concerns 65% of the studied mismatches and 60% of the domains with mismatching records. The mismatch often involves records from different protocols. We have observed that 67.6% of the nameserver mismatches were between a WHOIS record and an RDAP record, whereas

17% were between two RDAP records (registry RDAP and registrar RDAP) and 15.4% were between two WHOIS records of the same domain.

In summary, while affecting only 1% of domains, nameserver mismatches, especially the 67.6% involving disparities between WHOIS and RDAP, raise concerns. In 60% of such cases, both sources lack any common nameservers, making the choice between WHOIS and RDAP for gathering nameserver information non-neutral and yielding incompatible results.

Who is Right?

To successfully collect any DNS record for a domain it is essential to have an `NS` record in the parent zone file, specifying the authoritative nameserver for the domain. To gather the nameserver information, we actively queried the DNS infrastructure and performed a comparison with the nameservers listed in the WHOIS and RDAP records.

Methodology. To find the `example.com` nameservers, the client (e.g., a recursive resolver) first sends an `NS` query to the DNS root servers and receives the name of the servers that have authority over the `.com` zone. The client then sends another `NS` query to one of these servers and receives the `NS` record of `example.com`. This last answer comes from the registry in charge of the `.com` zone. The client can then either return the result because it retrieved the `NS` record of `example.com` from the authoritative nameservers of the parent (nameserver of `.com`) or perform additional `NS` queries to the nameservers received at the previous step and get the nameservers configured by the administrator of the domain. RFC 1034 [6] states that the nameservers returned by the registry and the nameservers configured by the administrator must be identical, but previous study [73] revealed that around 10% of the domains in the `.com`, `.org` and `.net` zones had differences between the nameservers provided by the parent registry servers and the nameservers provided by the child domain servers. If a domain is active, it must have an `NS` record at the registry level, as it is a part of the resolution chain. On the contrary, some domain owners do not put `NS` records in the child nameservers. To maximize the number of collected domains, we queried the `NS` resource records for each domain at the registry level.

Scans. To determine the consistency between registration data sources and DNS data, we used `zdns` [44] to retrieve the `NS` resource records of each domain where a mismatch was detected. Additionally, we collected their WHOIS and RDAP records for a second time, specifically between January 24th and January 27th, 2023. This ensured that all three data sources (DNS, WHOIS, and RDAP) were collected simultaneously, eliminating cases where domain configurations were altered during our scans.

While some domains had expired between our initial scan and this supplementary analysis, approximately 90% of the domains remained active and produced a `NOERROR` DNS response with non-empty results during the scan.

Results. The second data collection unveiled 365,521 distinct domains exhibiting nameserver mismatches.

After the collection of the new registration data and the `NS` records from the authoritative DNS servers, the resulting data falls into two categories: the mismatch can be between two records from the same protocol (two WHOIS records or two RDAP records), or between two records of different protocols.

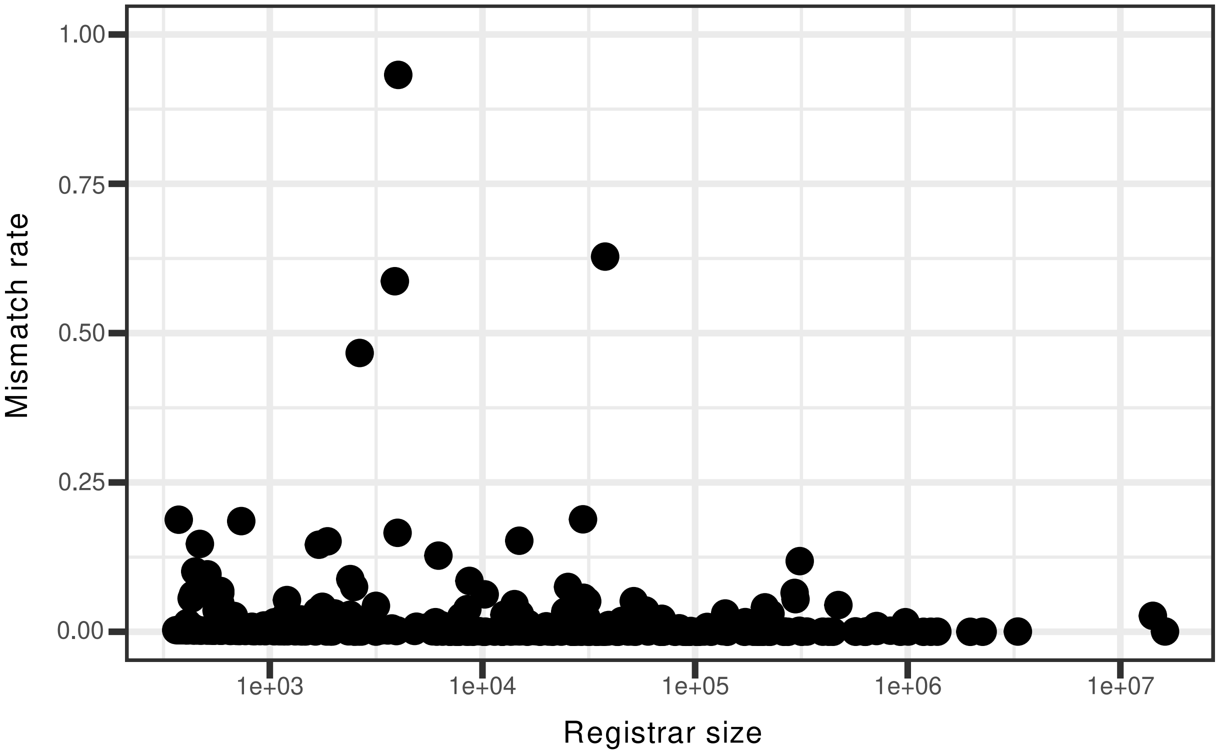


Figure 2.5: Nameserver mismatch rate per registrar

WHOIS-RDAP mismatches. In 74.9% of the identified mismatch cases, the disparity exists between a record gathered through WHOIS and a record collected through RDAP. As previously described, the nameservers obtained from DNS may constitute a subset, superset, or have a non-empty intersection with each record. Upon examining all possible scenarios, we found that in 99.5% of cases, the DNS record corresponded to either the WHOIS or RDAP record. The remaining 0.5% involved intermediate situations where the DNS result only partially matched one of the records. Due to the limited number of domains affected by this situation, we opted for concentrating our analysis on cases where the DNS matched one of the records.

In 78.5% of cases, the DNS data corresponded to the nameservers provided by the RDAP record. This underscores the fact that, although nameservers obtained from DNS typically align with data from RDAP, there are still 21% of mismatch instances where the DNS results match the WHOIS record. Interestingly, Figure 2.5 highlights that a few registrars exhibit a notably high mismatch rate compared to others. We observed that only four registrars have a mismatch rate exceeding 25%, while the largest registrars, representing the majority of domains, maintain a very low mismatch rate.

Registry-Registrar mismatches. The remaining 25.1% of cases represent the situations in which the mismatch is between two records from the same protocol but collected from different servers. In this case, the collector queried the registry server, got a referral to another server, and recursively called it, gathering an additional record. If two records are inconsistent, we checked if the nameservers provided by the DNS matched the records collected at the registry server or at the referral servers. In 99.2% of the cases, the DNS data matched the registry record, and in the remaining 0.8% of the cases, it did not match either records. The DNS data matched the registrar record in only 0.008% of the cases.

As described in Section 2.4.1, we decided to collect the NS records at the DNS authoritative nameservers of the registry. Consequently, we expected the record provided by the registry to be consistent with the DNS data from the same registry. Hence, the mismatches between two records from the same protocol almost always come from invalid data from the referral server.

The main takeaway is that when both sets of nameservers have no common elements, and the discrepancy lies between an RDAP and a WHOIS record, the RDAP record is accurate and aligns with the `NS` records from DNS in 78% of the cases.

2.4.2 IANA ID, Creation and Expiration Dates

When it comes to obtaining the IANA ID, creation date or registrar name of a domain, research primarily relies on the WHOIS and RDAP protocols. Unlike nameservers, which can also be retrieved from DNS, there is no third-party service that offers direct access to this data. Consequently, when two sources diverge in these fields, there is no simple method to determine which record contains the accurate information.

In this section, we outline the types of mismatches identified in IANA ID, creation and expiration dates, and highlight a few cases where we can ascertain the correct record.

Creation and Expiration Dates.

The creation date represents the domain's initial registration instant, providing insight into its age. In domain-related research, the domain age is a pivotal factor as older domains, active for multiple years, are generally deemed more trustworthy than newly registered ones. The extensive analyses of the domain registration behavior [65, 66] have shown that malicious domains tend to have shorter lifespans and are used in attacks shortly after registration. Other studies [64, 68] have used the creation date to detect bulk registrations of malicious domains.

The domain age is also frequently combined with other parameters to distinguish between benign and malicious domains [59, 67]. While some approaches [66] attempt to estimate the domain activity period by monitoring its appearance and disappearance in publicly accessible zone files, this method is contingent on zone file accessibility and the availability of historical data for the domain. Consequently, most studies depend on WHOIS or RDAP to acquire the creation date.

The expiration date also provides insights into the domain behavior and can shed light on various scenarios. For instance, if a domain is removed from its zone file before its expiration date, it may suggest actions taken by the registrar or seizure by authorities [66]. Additionally, parking and drop-catching entities use the expiration date to identify when a domain will become available for re-registration [60].

Both creation and expiration dates are usually found in the majority of WHOIS and RDAP records. However, in the case of WHOIS, they may be listed under various names, such as `Creation Date`, `Registration Date`, `Created at`, `Valid until`, and more.

After filtering out dates that were not possible to parse and dates lower or equal to the UNIX Epoch (which may indicate a default value or a configuration error), we observed that 5.7% (for creation dates) and 4.4% (for expiration dates) of the domains exhibited inconsistencies across their records. Figure 2.6 illustrates the distribution of time differences between these records.

We can observe that in 84% of the cases for creation dates and 78% of the cases for expiration dates, the differences are less than 2 days. These discrepancies have minimal impact on the analyses relying on creation dates to gauge the domain age [65] or on the speed of domain re-registration after expiration [66].

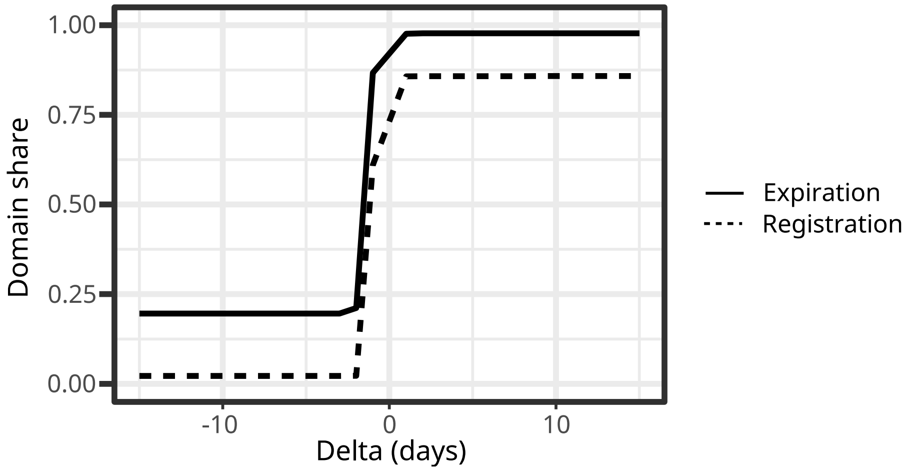


Figure 2.6: Cumulative distribution of creation and expiration date mismatches

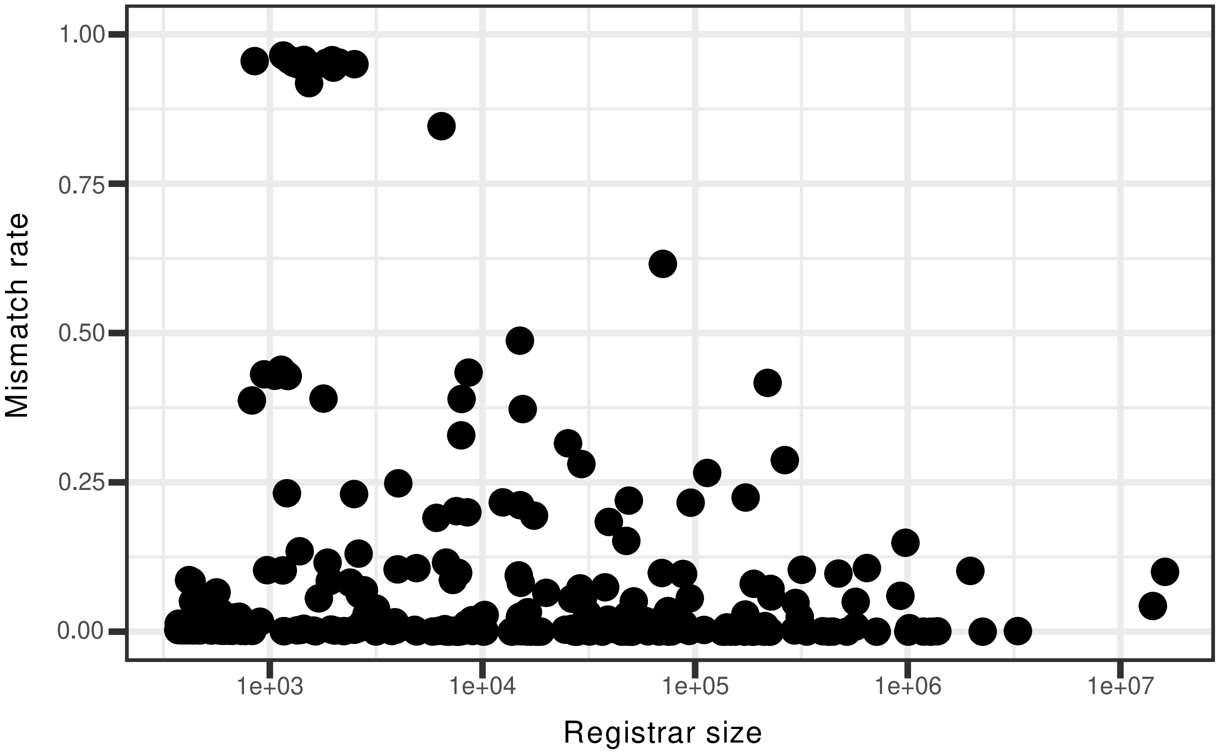


Figure 2.7: Creation date mismatch rate per registrar

Previous studies [47] highlighted common misunderstanding of the different expiration steps before the deletion of a domain and pointed out that these steps can account for a mismatch of up to 30 days, as a confusion could be made between the expiration date, the deletion date and how the grace and redemption periods should be accounted for, but the collected data shows no specific mismatch proportion at 30 days. However, our analysis points out that several records present an expiration date difference of exactly one year, which corresponds to the minimal duration of a registration, so the difference could come from the fact that the renewal of the domain was taken into account in one of the records and not in the other. Then, 98% of expiration date mismatches are either under 2 days or exactly 1 year, leaving only a few domains with unexplained expiration date mismatches.

Approximately 16% of the creation date mismatches extend beyond 2 days. In contrast to expiration date mismatches, creation date mismatches are more evenly distributed. One possible explanation for these discrepancies is that different entities may have distinct definitions of the `Creation Date`. While RFC 9083 [46] clearly defines keywords to describe creation events in RDAP, such as `registration`, `reregistration`, `reinstantiation`, and `transfer`, WHOIS lacks such precision. Consequently, the `Creation Date` recorded in the WHOIS record may not correspond to the same events in the domain life cycle as the `registration` event in the RDAP record.

The `Creation Date` mismatch rate for each registrar, as shown in Figure 2.7, highlights that while many registrars have over 10% of their domains with creation date mismatches, a few registrars exhibit nearly 100% of their domains with mismatched creation dates. This observation supports our hypothesis that some of these mismatches may result from registrar misinterpretations, custom registration processes or systematic configuration errors. For example, the vast majority of domains presenting a `Creation Date` mismatch of 30 or 31 days are under the `.com` TLD and share the same registrar, `FastDomain Inc`. For these domains, the registrar record `Creation Date` is always one month earlier that the one in the registry record. After investigation, we found that this registrar allows their customers to cancel their domain order up to 30 days after payment, while the ICANN Agreement [51] only imposes a 5-day refund window. Consequently, we can hypothesize that the creation of the registry record was delayed until the end of the 30-days period, while the registrar record was created when the customer first ordered the domain.

Table 2.4: Number of records and domains with mismatching emails

Case	Records	Domains
All	50.1M	19.0M
Inclusion	37.1M (74%)	15.1M (79.8%)
Intersection	0.59M (1.2%)	0.56M (2.9%)
Disjoint	12.4M (24.8%)	4.9M (26%)

IANA ID and Registrars.

ICANN-accredited registrars play a crucial role in domain registration and management. The IANA ID associated with each registrar is a unique identifier, often found in WHOIS and RDAP records, helping to trace domain ownership and authority.

The content of the Registrar field in WHOIS and RDAP may differ from the name listed in the IANA registry. For example, 2.4% of domains with IANA ID 146 (GoDaddy.com, LLC) have different Registrar entries, including GoDaddy LLC, GoDaddy.com, Inc., GODADDY or Go Daddy, LLC. Therefore, parsing the Registrar field to identify registrars can be challenging, and users often rely on the IANA ID for accuracy. However, in certain ccTLDs, registrars receive local accreditation, and the corresponding IANA IDs are not assigned or displayed in the public WHOIS and RDAP. In these cases, extracting registrar information relies solely on the Registrar field.

Our analysis uncovered that a mere 0.2% of domain names had records with inconsistent IANA ID. The analysis of IANA IDs reveals that the majority of mismatches occur between specific pairs of IDs. Approximately 91% of these detected mismatches involve a record with IANA ID 1556 (Chengdu West Dimension Digital Technology Co., Ltd.) and another record with IANA ID 1915 (West263 International Limited). Additionally, 4% of the mismatches involve IANA ID 3951 (Webempresa Europa, S.L.) and ID 5555555, which is an invalid ID. This pattern may suggest misconfiguration issues by particular entities, resulting in consistent mismatches across all the domains they manage.

In the second case, we confirmed the issue by registering a domain name with the registrar Webempresa Europa, S.L. and examining its records. While the registry WHOIS record correctly indicated the valid IANA ID 3951, the registrar WHOIS record contained an IANA ID field with the value 5555555, which does not correspond to any valid registrar number. The registrar's WHOIS record also displayed place-holder values for various fields, including the abuse contact phone number and the reseller name. We verified that all domains registered with this registrar had inconsistent records. We reported the issue to the registrar, and over several months, we noticed that all the domains they managed were updated with correct registration data, resolving the inconsistencies. We suspect that the mismatches between ID 1556 and ID 1915 share the same origin. However, we were unable to test this hypothesis, as both registrars exclusively serve users in China and Hong Kong.

2.4.3 Email Addresses

Various types of email addresses are included in registration data, serving different purposes. These addresses are associated with the registry, registrar, or registrant, as well as for technical, administrative, and abuse-related functions. RFC 9083 [46] provides specific keywords in RDAP for describing the role of each email address, such as administrative, abuse, billing, or technical. This allows for easy identification of the address role, a capability that WHOIS lacks.

For these reasons, we chose to collect all addresses in each record without distinguishing their roles. We then compared the records based on the sets of addresses they contain. Mismatches can occur due to protocol-specific contact addresses; for instance, the technical contact email for RDAP records may differ from that in WHOIS records if a registrar delegates technical administration to a third party. However, we anticipate that some addresses will be common across multiple records for the same domain, such as the abuse contact email for reporting domain-related abuse.

To analyze email mismatches, we applied the techniques described in Section 2.4.1. Initially, email addresses were parsed and duplicates were removed. Subsequently, we compared the various possible inclusion and intersection cases. The results of this analysis are presented in Table 2.4.

We identified 50 million mismatches on 19 million unique domains, encompassing 34.5%

of the domains in this study. Among them, 74% of mismatches and 79.8% of domains featured one set of email addresses included in the other. About 75.2% of mismatches were either inclusions or intersections, potentially arising from shared addresses (e.g., abuse or registrant emails) while the addition of server or protocol-specific addresses by different entities (e.g., contact addresses for WHOIS or RDAP servers) may result in differences. However, nearly 5 million domains (8.8% of all analyzed domains) had a pair of records with no common email addresses.

The disjoint cases may be attributed to the GDPR implementation. Previous research [49] explored the impact of GDPR on the availability of personal information fields before and after its enactment. Following the GDPR implementation, many registrars and registries replaced the registrants' personal details like the name, the phone number, and the email address in WHOIS and RDAP records with entries such as 'REDACTED FOR PRIVACY', effectively concealing this information. However, some entities introduced proxy email addresses to safeguard the registrants' actual addresses. These proxy servers mediate communication between proxy addresses and registrant emails. For example, in an RDAP record under the `registrant` role, one might encounter the address `b4ebaf9bfeba@withheld forprivacy.com`. While this conceals the registrants' personal data from the public, a valid contact address remains accessible. Protecting user privacy by redacting or using proxy email addresses can create discrepancies between WHOIS and RDAP records, as the registrant's address, which should be consistent in all records, may be redacted or hidden behind proxies.

Email mismatches can also occur when registrars or registries use distinct addresses for WHOIS and RDAP, even though both email addresses are administered by the same organization, such as `abuse.whois@registrar.com` and `abuse.rdap@registrar.com`.

To address these discrepancies, we conducted a new analysis by extracting and comparing only the domain names from email addresses, discarding the local parts. This approach considered email addresses within the same domain as consistent. The results are presented in Table 2.5. We found that this approach resolved 18.6% of the mismatches and reduced the rate of disjoint email addresses from 24.8% to 9.7%. This suggests that in many cases where email addresses appeared disjoint, they actually originated from records with different email addresses hosted under the same domain.

Table 2.5: Number of records and domains with email domain mismatches after removing the local part of the address, retaining only the base domain name

Case	Records	Domains
All	50.1M	19M
Equality	9.3M (18.6%)	4.0M (21.4%)
Inclusion	35.7M (71.3%)	14.5M (76.7%)
Intersection	0.24M (0.5%)	0.23M (1.2%)
Disjoint	4.8M (9.7%)	2M (10.6%)

In conclusion, this analysis underscores the need for caution when gathering email addresses, especially for notification campaigns [50]. The choice of data source significantly affects the collected email addresses for 34.5% of domains. Additionally, in 10% of cases where email records mismatch, the domains hosting these addresses are unrelated, suggesting that email servers may be managed by different entities, potentially leading to varying effectiveness in notification campaigns.

2.5 Related Work

Table 2.2 provides an overview of prior research that used WHOIS and RDAP data for domain name registration information. Nevertheless, the accuracy of the collected data has not been thoroughly investigated. Some earlier studies [47,63,64,68] relied on WHOIS data prior to the introduction of RDAP. However, as discussed in Section 2.2, inconsistencies are also present in WHOIS data obtained from servers managed by registries and registrars.

Challenges in processing WHOIS records have been identified, particularly concerning the reliability of extracted data such as AS numbers for IP WHOIS [74] and domain status [47]. In a previous in-depth analysis of the `.com` zone [61], the authors developed a machine-learning algorithm to address the multiple formats used in WHOIS records, demonstrating the difficulties in consistently parsing relevant fields.

The performance analysis of WHOIS and RDAP [48] focused on the speed but lacked the examination of data consistency across different servers and protocols.

In our work, we observed that 7.6% of the scanned domains exhibited mismatching records, raising concerns about the reliability of security metrics relying on such data. Notably, metrics that use the `Creation Date` field [67] or the bulk registration status [66] may be impacted, especially for registrars with high mismatch rates as presented in Figure 2.7. Obtaining accurate creation dates for domains under these registrars may require alternative data sources.

The `Emails` field exhibited the highest mismatch rates, even with a conservative parsing approach. Previous studies on notification campaigns [50,69] reported difficulties in extracting valid email addresses from WHOIS records, with email bounce rates exceeding 50%. These findings raise concerns about the effectiveness of notification campaigns due to the challenges associated with obtaining consistent and valid abuse emails from different entities.

2.6 Conclusions

Registration data plays a crucial role in the development of detection systems and gaining insights into the domain name behavior and entity management. However, obtaining this information may require interacting with various servers (either registries or registrars) and protocols (either WHOIS or RDAP). Our extensive analysis of 164 million records from 55 million domains unveiled that the data obtained through WHOIS and RDAP is generally consistent. Nonetheless, 7.6% of the analyzed domains displayed discrepancies in one or more of the following fields: IANA ID, creation and expiration dates, or nameservers. In cases related to the nameserver field, we used active DNS measurements to determine the accurate record. When disparities involved RDAP and WHOIS records, our findings showed that RDAP records were correct in 78% of instances where mismatches occurred.

The principal insight underscores the importance of studies reliant on dependable registration data to diversify their data sources by collecting it from various servers and protocols. Although larger registrars generally display lower mismatch rates, this observation does not inherently guarantee the accuracy of the data. Smaller registrars present a wide range of outcomes, with some demonstrating minimal discrepancies, while others exhibit higher rates. The potential risk exists for malicious actors to exploit registrars with inconsistent data, allowing them to evade detection systems that rely on the availability and reliability of registration data.

An analysis of the extent of malicious domains managed by such inconsistent registrars could offer valuable insights into evasion strategies.

To facilitate future research, we will provide the collected records (both raw and parsed) and the associated data analysis as resources linked to this publication.

3
Semantic Identifiers and DNS Names for IoT

In this chapter, we propose a scheme for representing semantic metadata of IoT devices in compact identifiers and DNS names to enable simple discovery and search with standard DNS servers. Our scheme defines a binary identifier as a sequence of bits: a Context to use and several bits of fields corresponding to semantic properties specific to the context. The bit string is then encoded as `base32` *characters and registered in DNS. Furthermore, we use the compact semantic DNS names to offer support for search and discovery. We propose to take advantage of the DNS system as the basic functionality for querying and discovery of semantic properties related to IoT devices. We have defined three specific Contexts for hierarchical semantic properties as well as logical and geographical locations. For this last part, we have developed two prototypes for managing geo-identifiers in LoRa networks, one based on Node and the Redis in-memory database, the other one based on the CoreDNS server.*

This chapter is based on a work published with Michele Amoretti, Fabrizio Restori, Maciej Korczynski and Andrzej Duda [1]

Contents

3.1 Introduction

Many IoT applications require the knowledge about the various properties of IoT devices that provide some data about the physical world and can act upon the environment. The properties may for instance include the information on:

- type and unit of data, (e.g., temperature in °C),
- resolution, frequency of data (e.g., 512 × 512 pixels every hour),
- possible actions performed by the device (e.g., switch on),
- raised alarms (e.g., overheating),
- geographic location of the device (e.g., $(+28.61, -80.61)$ WGS84/GPS coordinates),
- logical location of the device (e.g., Room 235 on Floor 14).

Several initiatives aimed at expressing and structuring this kind of IoT and M2M metadata: Sensor Markup Language (SenML) [75], OMA SpecWorks, and oneM2M Base ontology [76]. The World Wide Web Consortium (W3C) schemes for the semantic Web such as RDF,[1] OWL,[2] SPARQL[3] also allow understanding and discovery of IoT data. For expressing specific IoT semantics, W3C proposed a Semantic Sensor Network (SSN) ontology[4] that allows the description of sensors and their characteristics addressing the issue of interoperability of metadata annotations.

The Web of Things initiative of W3C[5] aims at unifying IoT with digital twins for sensors, actuators, and information services exposed to applications as local objects with *properties*, *actions*, and *events*. W3C Thing Description (TD)[6] expressed in JSON-LD[7] covers the behavior, interaction affordances, data schema, security configuration, and protocol bindings.

Thing Description allows for attaching rich semantic metadata to IoT devices, however, this format is oriented towards processing by non-constrained applications running for example in the Cloud or in the Edge to become the base for sophisticated discovery and search services offered on Web servers for IoT users and applications. However, we can notice that discovery and search based on semantic metadata also happens in constrained IoT environments where an IoT device needs to discover other devices and choose the right one for further communication or collaboration. In this case, semantic metadata of IoT devices need to be encoded in a highly compact way to reduce the overhead in usually bandwidth limited networks.

In this work, we propose a scheme for representing semantic metadata of IoT devices in compact identifiers or names to enable simple discovery and search with standard DNS servers. The idea of the scheme is inspired by the Static Context Header Compression (SCHC)[8] approach to IP header compression.

[1] http://www.w3.org/RDF
[2] http://www.w3.org/TR/owl-ref
[3] http://www.w3.org/TR/sparql11-query/
[4] https://www.w3.org/2005/Incubator/ssn/ssnx/ssn
[5] https://www.w3.org/WoT
[6] https://www.w3.org/TR/wot-thing-description
[7] https://www.w3.org/TR/json-ld
[8] https://tools.ietf.org/html/rfc8724

In SCHC, two devices that exchange IP packets compress headers based on pre-established contexts. Instead of a full header, a device inserts the information about the context to use and some short information required to reconstruct the header. In this way, a 40 byte IPv6 header can be compressed down to just a few bytes. Our scheme defines a binary identifier as a sequence of bits composed of a Context to use and several fields corresponding to semantic properties specific to the Context. The bit string is then encoded as `base32` characters and registered in DNS . Thus, the DNS name encodes in a compact form the semantic metadata of an IoT device.

We define several Contexts of identifiers expressing different semantic metadata to fit the most popular device characteristics (other can also be defined):

1. hierarchical semantic properties,
2. logical location of the device,
3. geographic location of the device.

The first one corresponds to the structured representation of the attributes of Thing Description and two others cover the geographical information about an IoT device. We instantiate the scheme for encoding geographic location in case of LoRa networks and show how to construct a 64 bit geo-identifier of LoRa devices.

Furthermore, we use the compact semantic DNS names to offer support for search and discovery. In constrained environments, providing full-fledged database search functionality may be difficult. Instead, we propose to take advantage of the DNS system as the basic functionality for querying and discovering the semantic properties related to IoT devices. Our encoding scheme of semantic metadata structures the DNS names similarly to IP prefixes: a longer prefix represents more specific information and shortening a prefix corresponds to more general information, thus allowing for some range or extended topic queries. For instance, if the name represents a geographical location, a longer name represents a smaller area and a shorter name corresponds to a larger zone that encompasses the smaller area designated by the longer name. Finally, we describe two prototypes supporting DNS queries on geo-identifiers.

Querying DNS based on semantic names can bring interesting features to many IoT applications: finding devices corresponding to a given property, placement on a map of all sensors belonging to a given application, sending commands to the devices in a chosen region, or gathering data from chosen devices based on their geographical location.

This work makes the following contributions:

1. we define a scheme based on Contexts for compact encoding of different types of metadata in DNS names,
2. we take advantage of geohashes to instantiate the scheme for encoding geographic location,
3. we propose a means for simple and minimal discovery of IoT devices and searching for their characteristics based on standard DNS functions,
4. we explore an idea of using DNS to store and publish IoT data,
5. we validate the proposed schemes with preliminary prototypes supporting DNS queries on geo-identifiers.

3.2 Related Work

We briefly review previous work related to expressing semantic properties of IoT devices and compact encoding of geographical location.

3.2.1 Semantic Properties of IoT Devices

As mentioned in the introduction, several initiatives considered the problem of expressing metadata of IoT devices and M2M communications: Sensor Markup Language (SenML) [75], IPSO Alliance Framework [77], and oneM2M Base ontology [76]. Kovacs et al. proposed a system architecture for achieving worldwide semantic interoperability with oneM2M [78]. The Semantic Sensor Network (SSN) ontology allows the description of sensors and their characteristics [79].

An important initiative of W3C aimed at creating the semantic Web of Things [80] to enable unambiguous exchange of IoT data with shared meaning. Previous work [81] discussed solutions that extend the Web of Things architecture to achieve a higher level of semantic interoperability for the Internet of Things. Nevertheless, many proposed approaches do not address the constraints of IoT devices that do not match the size and the form of semantic descriptions usually developed in the traditional W3C setting. For instance, previous studies [81] reported performance results coming from a testbed composed of two computers connected to an 802.11 network. Previous work on this topic led to the DINAS scheme [82], based on Bloom filters for creating compact names from node descriptions and a service discovery protocol for short-range IoT networks running RPL. Other work emphasizes the importance of DNS for IoT [83].

3.2.2 WGS84 aka GPS

WGS84 is a common format for encoding geographical coordinates used in GPS, composed of two numbers in degrees of the form `ddd.ddddddd`, where `d` stands for a degree digit. Degrees are expressed as numbers between -180 and $+180$ for longitude, and a number between -90 and $+90$ for latitude (locations to the west and to the south are negative), e.g., $(+28.61, -80.61)$ corresponds to the location of the Cape Canaveral Space Center.

Expressing a given geographical location is always done with a given precision, and when decoding a position, all methods return the center of the square representing all possible positions. For example, if we decode $(28°\text{N}, 80°\text{W})$, we know the position is in the square between $(28°\text{N}, 80°\text{W})$ and $(29°\text{N}, 81°\text{W})$, and we will return $(28.5°\text{N}, 80.5°\text{W})$ to minimize the error.

Table 3.1 represents the longitudinal resolution at the equator and at a latitude of $45°\text{N/S}$ with an increasing number of decimal figures and the corresponding number of bits to represent them. The idea is to relate the size of a region to the number of bits used for representing a given geographical coordinate and thus relate the size of a region to the size of an identifier. We can observe that 8 decimal figures encoded on 26 bits are sufficient to represent the location at the precision of around 1 m.

3.2.3 Geoprefixes, Geohashes, Plus Codes

Previous work [84], defined the notion of a *geoprefix* for IPv6 networks: the location of each device is encoded in its IPv6 multicast address and an application can send a packet to all devices corresponding to a given prefix representing a geographic area (a geocast).

[85] proposed *geohash*, an encoding of WGS84 coordinates based on Morton codes [86] that computes a 1-dimensional value from the 2-dimensional GPS coordinates by interleaving the binary representations of the coordinates and then encoding the result as ASCII characters.

Table 3.1: Longitudinal decimal degree precision

# of figures	# of bits	Equator	45°N/S
3	9	111.3200 km	78.710 km
4	12	11.1320 km	7.871 km
5	16	1.1132 km	787.100 m
6	19	111.3200 m	78.710 m
7	22	11.1320 m	7.871 m
8	26	1.1132 m	787.100 mm

Table 3.2: Combining latitude and longitude encoded in a unique binary value

Longitude	0111
Latitude	1011
Result	01101111

In this method, to encode a given location, we proceed by a dichotomy. Starting with the full interval ($[-180;+180]$ for longitude, $[-90;+90]$ for latitude), we split the interval in two ($[-90;0]$ and $[0;+90]$ for latitude), then, if the location is in the higher half, we add bit 1 to the encoding of the coordinate, or else, we add bit 0, and we repeat the operation with the new interval, building the encoding bit by bit, until we reach the desired precision. When decoding, we start with the first bit, and reduce the area according to the value of each bit. Once the last bit is reached, the decoded location is at the center of the remaining interval (for example, for latitude, if we have only one bit with value 1, we would decode that the latitude is $+45$, the middle of the upper $[0;+90]$ interval). With this method, each additional bit halves the size of the interval and doubles the precision.

Once both latitude and longitude are represented this way, their binary codes are intermingled to produce a unique value: odd bits represent latitude and even bits represent longitude as presented in Table 3.2. For example, the resulting encoding of latitude 1011 and longitude 0111 is 0110 1111.

Geohash-36,[9] originally developed for compression of world coordinate data, divides the area into 36 squares and generates a full character from a set of 36 predefined characters describing which sub-square contains the position.

Google Maps uses *Plus Codes* [87, 88] made up of a sequence of digits chosen from a set of 20. The digits in the code alternate between latitude and longitude. The first four digits describe a one degree latitude by one degree longitude area, aligned on degrees. A Plus Code is 10 characters long with a plus sign before the last two:

1. The first four characters are the area code describing a region of roughly 100 × 100 kilometers.

[9] https://en.wikipedia.org/wiki/Geohash-36

Table 3.3: Longitudinal decimal degree precision and the size of a geohash

length	lat bits	lng bits	lat error	lng error	error
1	2	3	± 23°	± 23°	± 2500 km
2	5	5	± 2.8°	± 5.6°	± 630 km
3	7	8	± 0.70°	± 0.70°	± 78 km
4	10	10	± 0.087°	± 0.18°	± 20 km
5	12	13	± 0.022°	± 0.022°	± 2.4 km
6	15	15	± 0.0027°	± 0.0055°	± 610 m
7	17	18	± 0.00068°	± 0.00068°	± 76 m
8	20	20	± 0.000085°	± 0.00017°	± 19 m
9	22	23			
10	25	25			± 59 cm
11	27	28			
12	30	30			± 1.84 cm

2. The last six characters are the local code, describing the neighborhood and the building, an area of roughly 14 × 14 meters.

As an example, let us consider the Parliament Buildings in Nairobi, Kenya located at the `6GCRPR6C+24` plus code: `6GCR` is the area from `2S 36E` to `1S 37E`. `PR6C+24` is a 14-meter wide by 14-meter high area within `6GCR`. The + character is used after eight digits, to break the code up into two parts and to distinguish codes from postal codes.

3.3 Compact Encoding of IoT Metadata

The main objective of this work is to design a scheme for encoding semantic properties in DNS names so that IoT devices could discover relevant nodes using with DNS name resolution. Figure 3.1 gives an example of how it can be done in the context of LoRa devices. Note that DNSSEC guarantees the information integrity.

We propose to assign *self-certifying names* to IoT devices: the name derives from a public key to enable secure establishing of the identity of a device without relying on an external PKI infrastructure. The self-certifying name is constructed as a hash of public key K_p similarly to Bitcoin addresses:

$$A = ripemd160(sha256(K_p))$$

then A is encoded with `base32` (20 characters) giving the DNS name N. `base32` encoding represents a binary string with `0-9` digits and some lower case letters (excluding characters hard to distinguish like `i`, `l`, `o`). We cannot use `base58check` like in Bitcoin because DNS names do not distinguish between upper case and lower case and `base58check` contains those different

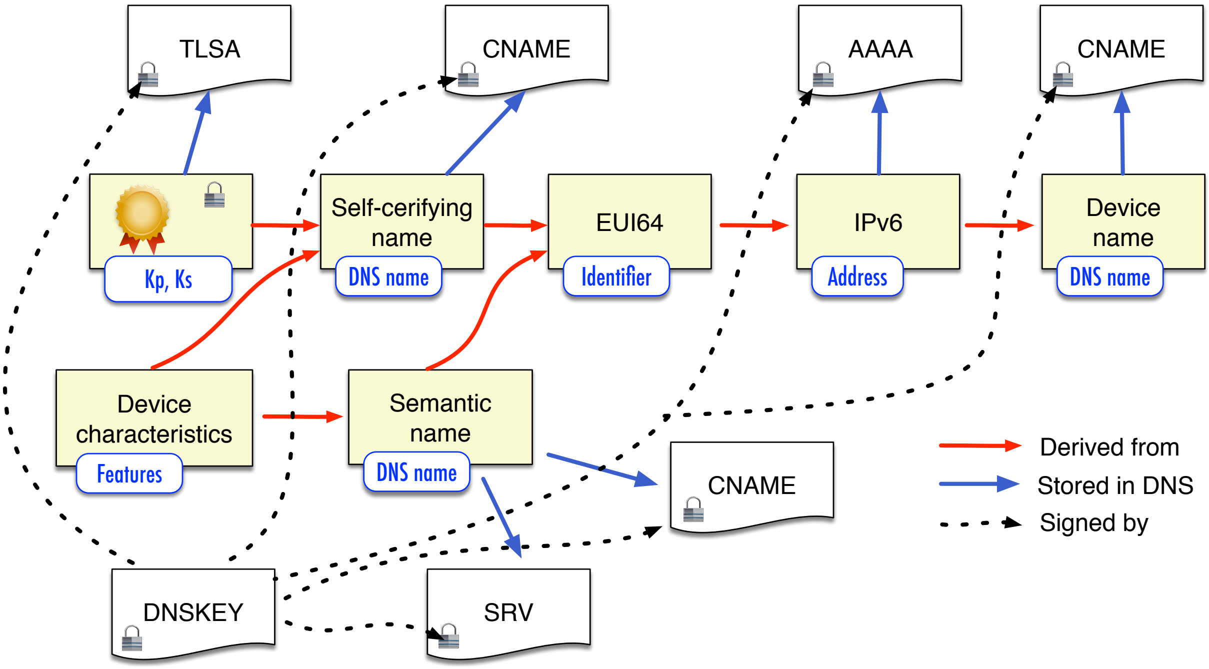


Figure 3.1: General scheme for identifiers and names.

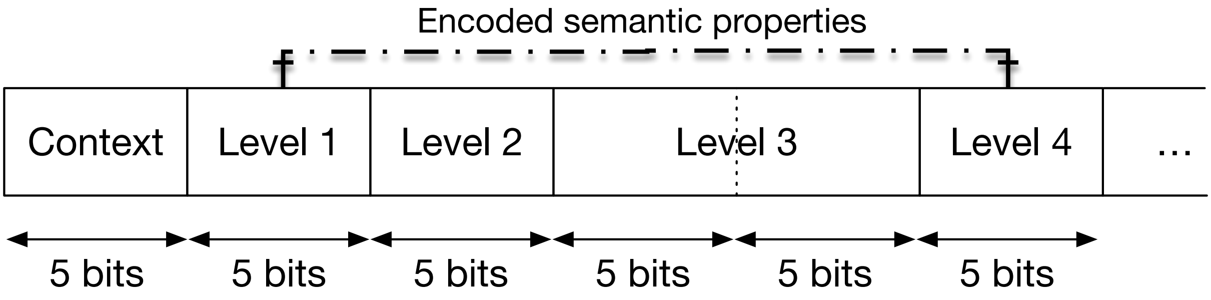


Figure 3.2: Structure of a binary semantic identifier (fields of 5 bits or a multiple of 5 bits).

versions. The nice feature of this scheme is that devices can check whether a public key from the TLSA DNS record corresponds to the name and if authentication is enforced (signature with the private key K_s) to be sure a device communicates with the right peer.

Then, we can derive an 8 byte `EUI64` identifier from A with SHA-3(A). `EUI64` identifiers are required in some networks like LoRa—we can obtain the LoRa `DevEUI` identifier derived from K_p and then use it to construct an IPv6 address. In the following sections we will show that the `DevEUI` of a LoRa device can represent its geographical location.

In addition to the self-certifying name, we will define other names (DNS aliases) that represent device properties encoded in a compact way. Moreover, we want the encoding scheme to take advantage of some discovery functionalities of DNS by requiring that a name is structured as an IP prefix—smaller prefix means a more general query.

3.3.1 Encoding Hierarchical Semantic Properties

Figure 3.2 presents the structure of an identifier. To decode an identifier, we first read the Context encoded in the first 5 bits, this tells us how to decode each property encoded in the rest of the identifier (called Level 1, Level 2, etc.).

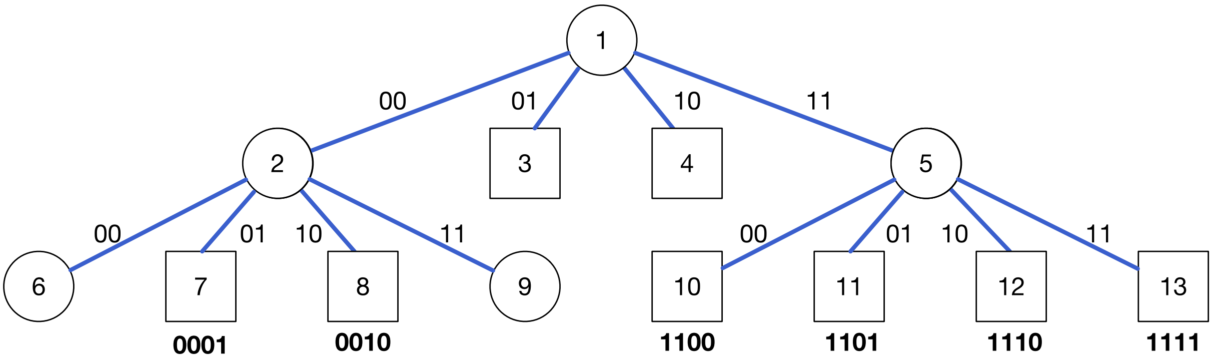


Figure 3.3: Semantic attributes encoded as a quadtree.

The first type of Context we will describe is where each property is encoded by a semantic tree with leaves corresponding to the different possible values of the property. Each property may use a different tree, the Context defines which tree to use for each field in the identifier. In this situation, the value of each property is encoded as the binary code generated when traversing the tree from the root to the value. Figure 3.3 presents an example of a semantic tree of degree $n = 2^2 = 4$ (where each branch of the encoding uses two bits) where non-terminal nodes are represented with a circle and property values are represented with squares. The position in the tree determines the code of a value, for instance, the value at leaf 12 has the code of `1110` corresponding to the traversal of the `11` branch and then, the `10` one, landing on 12. For each property, we first get its tree and the property length from the Context. Then, we traverse the tree based on the encoded binary value. If we land on a leaf (a value), means that the device's property is this value (for example, if the property is *Unit of the returned reading*, a value may be *kilograms*). If we land on a non-terminal node, it is a way to describe all devices whose property has a value in the subtree (for example, if the property is *Country of the device*, the first depth level of the tree may be the continents, so stopping at the first level represents all countries in this continent). For simplicity, in this example tree, all properties are encoded with a multiple of 2 bits, but in our design, we use 5 bits to encode each branch in a tree. This way, each $2^5 = 32$ possible step can be encoded as a `base32` letter, so each property value is encoded with a multiple of 5 bits. This choice avoids problems of dealing with padding if the size of the binary identifier is not a multiple of 5 bits.

Once the values of the different properties are encoded in `base32`, the resulting string can be used as a name for the device, allowing all users that know the name of the device to determine its properties.

3.3.2 Encoding Logical Location

In many use cases, an IoT application may benefit from metadata about localization in a logical form. For instance, when defining group communication for the Constrained Application Protocol (CoAP), RFC 7390 [89] considered a building control application that wants to send packets to a group of nodes represented by the following name:

`all.off376.floor1.bldg6.example.com`.

Logically, the group corresponds to *"all nodes in office 376, floor 1, building 6"*. Such hierarchical groups of fully qualified domain naming (and scoping) provide a logical description of places that may complement other precise geographical information that we will consider in the next section.

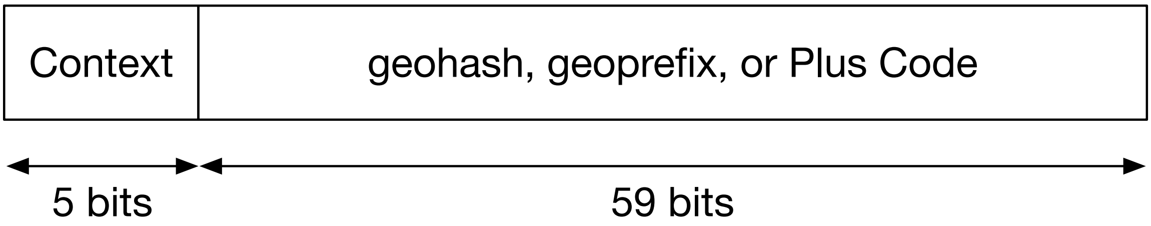


Figure 3.4: Structure of a geo-identifier on 64 bits.

We can observe that there is an inclusion relationship between elements of the description: office 376 is on the `floor 1`, inside `building 6`. A specific Context can represent this inclusion relationship. Assuming that we have up to 32 buildings, 32 floors per building, and 1024 rooms per floor, to encode the location of Room 376 on Floor 1 in Building 6, we define the binary identifier composed of the following fields (`base32` encoding in parentheses):

- `00010` (2) - Context-2
- `00110` (6) - Building 6
- `00001` (1) - Floor 11
- `01011` (`c`) `11000` (`s`) - Room 376

In this example Context-2 defines the first two properties to be the building and the floor, encoded on 5 bits each, and the third property to be the room number, encoded on 10 bits, as in Figure 3.2. The binary identifier `00010 00110 00001 01011 11000` results in the `261cs` identifier.

3.4 Encoding Geographic Location

Many IoT applications require precise information on the geographical location of IoT devices—when a sensor provides some measurement data, one of the most important additional information is the localization of the data source, usually stored as metadata. We can use GPS for localization, however, adding GPS to an IoT device increases its cost and energy consumption, which may make their cost prohibitive for many large scale IoT applications.

We take the example of LoRa networks to consider the problem of representing geographical locations in identifiers and DNS names. We propose a scheme to define the *geo-identifier* of a LoRaWAN device in a way that encodes its geographical location. Figure 3.4 presents its structure with two fields: 5 bits for the Context and 59 bits for encoding geographical coordinates of different forms. The Context gives the information about the type of encoding used in the remaining 59 bits.

LoRaWAN defines `DevEUI`, a unique 64 bit identifier in the IEEE EUI-64 [90] configured on a device. In the activation process of the device, it obtains a `DevAddr`, a 32 bit identifier in the current network allocated by the Network Server. In the situation where the device location is known at registration time, we propose to use a geo-identifier as `DevEUI`, store it as a DNS name, and provide a lookup service based DNS service discovery that returns names corresponding to a geographical region. With this setup, knowing the name of the device gives us information about its location, and querying devices based on their names allows to query them based on their location.

Table 3.4: Practical example of a geohash

geohash	Latitude	Longitude
`dr5r7p4rx6kz`	40.689167	-74.044444
`dr5r7p4`	40.69	-74.04
`dr5r111`	40.61	-74.13

With 59 bits for encoding the latitude and the longitude, a geoprefix or a geohash will result in a resolution of a few cm. Table 3.3 presents the size of the `base32` encoded geohash, the number of bits representing longitude and latitude, and the precision of the decoded value.

Geohashes offer interesting features: i) similar geohashes represent nearby positions and ii) a longer geohash represents a smaller area and shortening it reduces the precision of both its coordinates to represent a larger region.

Table 3.4 presents a practical example of the prefix property of a geohash. In this table, the second geohash is a prefix of the first one, so the area described by the first hash is more precise and is inside the area described by the second hash. The second and the third geohashes have a common prefix, so they are in the same region (easily computable with the common prefix) but do not overlap.

Plus Codes can be shortened relative to a reference location, reducing the number of digits to use to reach a given precision. In our schema, the reference point could be defined in the Context, allowing high geographical precision because similarly to geohashes and geoprefixes they represent areas and the size of the area depends on the code length.

We can store a string version of the geohash or Plus Code in DNS as the names of an IoT device and enable some geographical/proximity searches using the `geohash.org` site or Google Maps (for Plus Codes).

The only constraint of using geo-identifiers for `DevEUI` is the fact that `DevEUI` does not have the EUI-64 format anymore, which may be an obstacle for some applications. On the other hand, we gain the possibility of linking the device location with its identifier.

3.5 Device Discovery with DNS Queries

In constrained environments, providing full-fledged database search functionality may be difficult. Instead, we propose to take advantage of the DNS system as the basic functionality for querying and discovering the semantic properties related to IoT devices. In this section, we discuss how to query the DNS system to discover the properties of IoT devices or find devices with given properties.

3.5.1 DNS Service Discovery

DNS-Based Service Discovery (DNS-SD) [91] is a functionality of DNS to discover services in a network. Information about a given service is stored in the DNS database as an `SRV` record of the form:

```
<Instance>.<Service>.<Domain> IN SRV <data>
```

and gives the target host and the preassigned port at which the service instance can be reached. The TXT record for the same name may give additional information about this instance in a structured form using key/value pairs.

A DNS client can discover the list of available instances of a given service type using a query for a DNS PTR record with a name of the form <Service>.<Domain> which returns a set of zero or more PTR records giving the <Instance> names of the services that match the queried <Service>. Each PTR record is structured as such:

```
<Service>.<Dom> IN PTR <Instance>.<Service>.<Dom>
```

The <Instance> portion of the PTR data is a user-friendly name consisting of UTF-8 characters, so rich -text service names and subdomains are allowed and encouraged, for instance:

```
LoRa temp sensor.Room 7._iot._udp.example.com.
```

The <Service> portion of the query consists of a pair of DNS labels, following the convention already established for SRV records, for instance, the PTR entry for name _http._tcp.local.:

```
_http._tcp.local. PTR web-page._http._tcp.local.
```

advertises a “web-page” accessible over HTTP/TCP.

We propose to use this mechanism for querying DNS to find devices relevant to properties or locations expressed in as our semantic names. For example, if Context 3 describes that the rest of the identifier is a geohash, the following query:

```
_3dr5r7p4r._iot._udp.iot.org IN PTR
```

would look for IoT devices near the Statue of Liberty, as the 3dr5r7p4r sub-domain can be decoded as:

- Context 3 (the first base32 character 3)
- Geoprefix dr5r7p4r, that translate to the location of the Statue of Liberty

3.5.2 Structuring Queries as Subdomains

The DNS system stores resource records in a hierarchical tree in which servers can delegate the management of subdomains. For example, the authoritative DNS server of example.fr can delegate the management of data.example.fr and all its entries and subdomains to another DNS Server. In a similar way, we can delegate the management of a given geographical region to a specific server whose region is included in the encompassing region of the delegating domain. For instance, if we want to delegate the management of the New York area to a city-managed DNS server, we could define a “New York area” subdomain and delegate it.

The in-addr.arpa domain uses this kind of method to delegate the management of an IPv4 address to the owner: when making a reverse DNS query on 1.2.3.4, the user queries 4.3.2.1.in-addr.arpa, the in-addr.arpa server delegated the 1.in-addr.arpa subdomain to the managers of 1.0.0.0/8, who in turn delegated the management of 2.1.in-addr.arpa to the managers of 1.2.0.0/16 and so on.

We can use a similar method to split semantic names into multiple subdomains to easily delegate some properties or locations to other servers. Here is an example for geo-identifiers:

instead of having to encode all possible geo-identifiers under the `_iot ._udp.iot.org` domain, we create the `dr._iot._udp.iot.org` subdomain and let it handle all areas with the `dr` geoprefix (encompassing the east coast of the USA). The server in charge of the `dr` prefix (east coast) can then delegate the `dr5r` area (encompassing New -York) to another server (a city managed server for example) by delegating the `5r.dr._iot._udp.iot.org` zone to the city managers. Then, the administrator of this server can choose to handle the `7p.5r.dr` subdomain itself, as it represents an area of 600 meters around the Statue of Liberty.

As a result, instead of querying `dr5r7p._iot._udp.iot.org`, we can query `7p.5r.dr._iot._udp.iot .org` and let each subdomain administrator choose if they want to delegate some sub-areas to other servers.

There are several ways to split a given semantic name into multiple subdomains so the user has to know the number of characters in a given subdomain to use it in a query. The number of bytes in each subdomain also influences the kind of queries a user can do. For example, setting 2 characters per subdomain, like in `34.12._iot._udp.iot.org`, makes it impossible to query directly for devices with the `123` prefix, so the user has to either query the whole prefix `12._iot ._udp.iot.org` and then filter the relevant results, or query all `3[0-f].12._iot._udp.iot.org` domains (16 queries). Thus, we need to choose the subdomain size carefully We propose three schemes for splitting geo-identifiers: a static subdomain length, a dynamic subdomain length, and multiple subdomain lengths.

Static subdomain length. We set size S for all subdomains. In this way, the user can split the geo-identifier in several groups of size S (rounded up or down, depending on the preference of a query on the encompassing zone and then filtering, or making multiple sub-queries) without any additional knowledge. The drawback is the lack of flexibility and the arbitrary choice of S that may be suitable for a given area but not for another one.

Dynamic subdomain length. Each domain has a `TXT` record that gives the size of the subdomains related to a given area. For example, `12._iot._udp.iot.org IN TXT "len=3"` informs the user that under the `12` subdomain, each subdomain has length 3, so one can query `345.12._iot._udp.iot.org`. This scheme supports the right subdomain length for each region: in a dense area where we need multiple precise subdomain delegations, we can set a small length to obtain precise subdivision and in sparse areas where we do not need small subdivisions (seas, fields), we can use a larger length. The scheme supports multiple subdomain lengths in the same query as in `6.345.12._iot._udp.iot.org` as each subdomain can set its size. The drawback is the need to recursively query different subdomains for their `TXT` records to know the length of each field before splitting the query the right way.

Multiple subdomain lengths. There are multiple ways to get to a given subdomain, so multiple ways of splitting the geo-identifier are possible and valid. For example, both `345.12._iot._udp.iot.org` and `5.34.12._iot._udp.iot.org` are valid and point to the same area. In this way, the users do not have to query for `TXT` records and can split their queries as they want. However, it may be hard to encode all ways of splitting the geo-identifier into subdomains in a resource record.

We can simplify this method with `CNAME` records, the same way the `in-addr.arpa` domain handles the delegation of subnetworks with arbitrary size[10] by defining multiple `CNAME` records. For example, if two different servers need to handle the prefixes `12a` and `12b` but the `12._iot._udp .iot.org` domain only defines subdomains of length 2, we can insert the following records:

[10] https://tools.ietf.org/html/rfc2317

```
a NS server.handling.a.12.area
a0 CNAME 0.a.12._iot._udp.iot.org
a1 CNAME 1.a.12._iot._udp.iot.org
a2 CNAME 2.a.12._iot._udp.iot.org
...
af CNAME f.a.12._iot._udp.iot.org
```

We can apply the same approach to all 16 bX.12 records. Once the CNAME records are created, a user querying a2.12._iot._udp.iot.org will be redirected to 2.a.12._iot._udp.iot.org, so they will try to resolve the a.12 part and will receive an NS entry pointing to the server in charge of the 12a area. Therefore, with these records, the user does not have to know how the delegation in the 12 area works, the query does not change from their point of view, but with CNAME and NS records, we can transparently delegate parts of the subdomain. Moreover, this method allows for easy modification of the server authoritative for a.12 because changing the NS entry is easy and the CNAME records remain the same. The method may generate many CNAME entries, but they are simple to generate automatically and do not need to change often.

Splitting into different subdomains can also apply to different contexts like for logical localizations. In this particular case, we can easily encode the properties in different subdomains because they are naturally ordered (a room on a given floor in a given building). For example, if the Context for Logical Localization is 2, the position of a device in Building 1 on Floor 5 in Room 56 is as follows (with base32 geohash in the parenthesis):

- Context-2: 2 - (2)
- Building: 1 - (1)
- Floor: 5 - (5)
- Room: 56 - (1s)

So, to get the sensors in this room, we send the following query:

```
1s.5.1.2._iot._udp.iot.org IN PTR
```

3.5.3 Use of AXFR for the Result Set

Another way of obtaining the result set from a DNS server is to use the DNS Zone Transfer Protocol (AXFR) [92] that returns all records in a zone. When a client sends an AXFR query message to an authoritative server, it answers with all resource records stored in the zone. Not all servers answer an AXFR query, as it requires good bandwidth, but we can take advantage of this feature to return the results of a query on subdomains describing a geographical area small enough so that the number of devices is reasonable. This feature can be used to return the results of a query on subdomains describing a property or a geographical area of the interest. For instance, to get all devices and the corresponding data stored in the zone in 123456, the user can use the following command:

```
dig AXFR 56.34.12._iot._udp.iot.fr
```

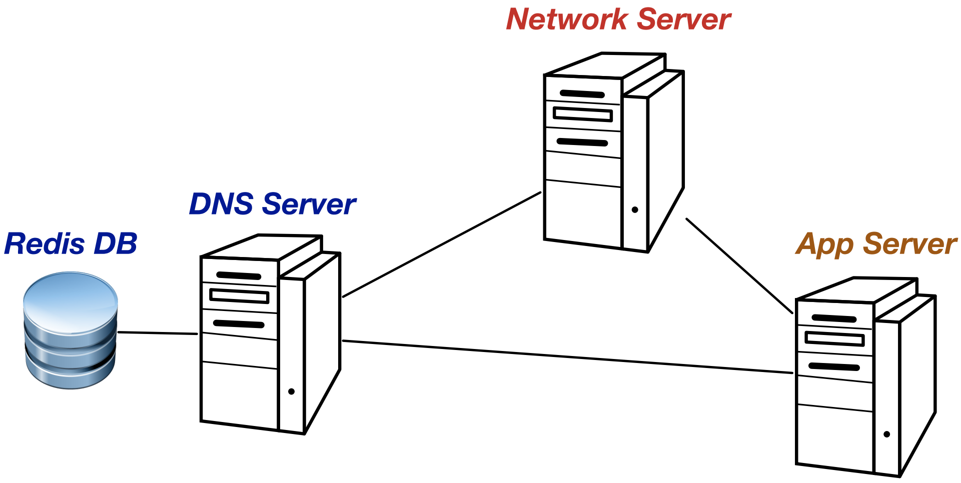


Figure 3.5: Prototype for LoRa geo-identifiers based on DNS-SD.

3.6 Prototype Implementation of Semantic Discovery

We have implemented two prototypes for geo-identifiers of LoRa devices. Their extension to consider other types of semantic names is undergoing. The prototypes for geo-identifiers are available to the public to encourage reproducibility. We show below some examples of their utilization with commands using the `dig` tool. These prototypes only consider geohashes encoded in the domain name without the use of the Context described in Section 3.3.

The first prototype[11] which takes advantage of the Node-based *dns2* module [93] and the Redis in-memory database,[12] allowed us to quickly deploy and test the concepts based on hard-coded data. Its general architecture is described in Figure 3.5.

The second prototype[13] uses the CoreDNS DNS server[14]. CoreDNS is highly flexible thanks to plugins that perform different functions: DNS, Kubernetes service discovery, Prometheus metrics, rewriting queries, and many more. We modified the *file* plugin that enables serving zone data from an RFC 1035-style master file.

In our prototypes, Applications or Network Servers that want to discover the location of LoRa devices can query a DNS server to find the devices matching some criteria based on their location. In other types of networks, devices themselves can directly query a DNS server.

In a LoRa network with geo-identifiers, when registering a device, the Network or Join Server registers several records in the DNS database. First, an `SRV` record giving the domain and ports where the Network Server managing a given device can be queried. Then, `PTR` records that allows finding the device based on its geo-identifier or name:

```
<name>._iot._udp.<Domain> IN SRV <port> <domain>
_<geo-i>._iot._udp.<Domain> IN PTR <name>
```

`<name>` being the semantic name like described in Section 3.3. This name of the given domain is unique and describes the properties of the device. `<geo-i>]` is the geo-identifier of the device encoded in multiple subdomains as described in Section 3.5.2. When an application

[11] https://github.com/dsg-unipr/geo-dns
[12] https://redis.io
[13] https://github.com/fabrizior/coredns
[14] https://coredns.io

needs to find all devices in a given area, it can query DNS for all devices in the matching subdomain by sending a query like:

```
_<geo-i>._iot._udp.<Domain> IN PTR,
```

where <geo-i>] can be split into multiple subdomains if needed.

The DNS server answers with the list of all PTR records in the queried subdomain, and therefore, in the represented area. Each PTR record gives the semantic name of a device in the area. Once the application knows the name of the devices in the area, it can query the DNS server for an SRV record with the different semantic name and get the Network Server managing the devices.

We have implemented this method in our prototypes and both of them can be queried with the dig tool[15] Upon receiving a PTR query for a specific <Service>, the server returns all instances of that service type in the subdomain:

```
# dig @127.0.0.1 -p 53 _dr._iot._udp -t PTR
;; QUESTION SECTION:
;_dr._iot._udp. IN PTR

;; ANSWER SECTION:
_dr._iot._udp. 100 IN PTR humidity.dr12._iot._udp.
_dr._iot._udp. 100 IN PTR temperature.dr34._iot._udp.
_dr._iot._udp. 100 IN PTR temperature.dr56._iot._udp.
```

Then, once the application has obtained the semantic name of the device (for example, temperature.dr56), it can query the server for an SRV record with this name, which will contain the domain and ports at which access the device. It can also ask for TXT records to get additional data about the device. For example, still using dig:

```
# dig @127.0.0.1 -p 53 temperature.dr56._iot._udp -t ALL
;; QUESTION SECTION:
;temperature.dr56._iot._udp. IN ALL

;; ANSWER SECTION:
temperature.dr56._iot._udp. 100 IN SRV 10 20 8080 dr56.unipr.it.
temperature.dr56._iot._udp. 100 IN TXT "temperature=14, unit=C"
```

Finally, when an A query for the <Domain> managing a device is received, the server returns the IP address of the Network Server the LoRa device is associated with.

For example:

```
# dig @127.0.0.1 -p 53 dr56.unipr.it -t A
;; QUESTION SECTION:
;dr56.unipr.it. IN A
;; ANSWER SECTION:
dr56.unipr.it. 100 IN A 160.78.28.203
```

[15] The address 127.0.0.1 used in the following examples should be replaced with the actual IP address of the DNS server.

3.7 DNS as a Source of IoT Data

In the previous sections, we have presented the schemes for encoding device properties in domain names to discover devices by querying the DNS infrastructure. Once the user discovers some relevant devices, they still needs to contact them with different protocols to obtain data or set up data delivery process with the COAP Observe option for instance. We can also take advantage of the DNS infrastructure as a public store for IoT data in a similar way to the Cloud. Many IoT applications store data in the Cloud for further processing and access by clients.

The idea of DNS as a source of IoT data is to use a `TXT` record associated with a name of an IoT device to store its data so that a large number of users can access the data in DNS instead of getting them directly from the device. As a `TXT` record linked to a domain is usually already filled with human-readable data related to the domain, we can add dynamically created records. Once the IoT data is stored in the `TXT` record, users will benefit from the DNS caching infrastructure efficient dissemination: recursive resolvers will cache its content and keep the data until the time-to-live (`ttl`) of the record expires. Then, the recursive resolvers will query the authoritative server to get the new record and the updated data from the device. With this method, the end users do not need to know what kind of protocol should be used to contact the device, as data is stored in a standard `TXT` record and no direct contact between the user and the device is required.

3.7.1 Encoding Data in TXT Records

RFC 6950[16] describes under what conditions an application can use DNS to store data and provides several recommendations and warnings indicated by other RFCs. RFC 1464[17] formalized the `<key>=<value>` format for storing data in `TXT` records, so in the case of the example of a temperature sensor, the DNS entry could be `<domain> IN <ttl> TXT "temperature=14"`.

Not all types of data should be placed in DNS: records with a large size can be used by attackers as an amplifier to generate a lot of traffic [36] (this is why `.com` records are limited to 1460 bytes). Therefore, this solution may not be suitable for all kinds of sensors. For example, a device taking periodic 512×512 pictures would generate data that should not be put on DNS, instead, the user will have to find a way to contact the device or its Network Server to get the data from a suitable source.

3.7.2 Updating Data in TXT Records

To keep data in the DNS record updated, there should be a process or an entity that gets the data from the device and updates the corresponding `TXT` record. For non-constrained devices, an IoT device could update its own record, but for most constrained devices, this kind of operation may be too costly, so another entity should update the data. For LoRa networks, all data from the devices go through the Network Server. As this server is not constrained, it can update the `TXT` record on behalf of the device using, for example, secure Dynamic DNS Update protocol extension [31] and a standard Unix `nsupdate` command to insert the new values in the zone file of the authoritative DNS server.

Because the data is not updated in real time, it is important to choose a suitable `ttl` value for the `TXT` record, so that when cached by recursive resolvers or proxies, the entry is marked as out

[16] https://tools.ietf.org/html/rfc6950

[17] https://tools.ietf.org/html/rfc1464

of date when new values are available. The ttl value must take into account the frequency at which the Network Server retrieves the new data from the device and updates the corresponding DNS record. For example, if the Network Server retrieves the temperature data and dynamically updates TXT records every hour, then the ttl value should also be set to one hour so that the information stored in caches of local DNS resolvers, which request the data on behalf of local clients, gets updated.

We can also use the Incremental Transfer mechanism (IXFR)[18] designed to transfer only a modified part of a zone, for example, the updated TXT records with the changed temperature. Each time the zone is dynamically updated by, for example, the Network Server, the serial number of its zone is increased. Therefore, after the initial AXFR transfer, the client should keep record of the Start of Authority (SOA) serial number of the transferred zone. Next, the client can send an IXFR request with the registered version number so that the authoritative name server responds only with the deleted and added resource records since the version known by the IXFR client up to the current version of the zone stored by the authoritative server. For example, to get new data related to the 123456 location, the client can use the following command:

```
dig @server IXFR=[old-ser] 56.34.12._iot._udp.iot.fr
```

3.8 Conclusion

In this work, we have proposed a scheme for representing semantic metadata of IoT devices in compact identifiers and DNS names to enable simple discovery and search with standard DNS servers. Our scheme defines a binary identifier as a sequence of bits composed of a Context and several bits of fields encoding semantic properties specific to the Context. The bit string is then encoded as a character string, stored in DNS. In this way, we may take advantage of the DNS system as the basic functionality for querying and discovery of semantic properties related to IoT devices.

We have defined specific Contexts for hierarchical properties as well as logical and geographic locations. For this last part, we have developed two prototypes that manage geo-identifiers in LoRa networks to show that the proposed scheme can take advantage of the standard DNS infrastructure.

[18] https://tools.ietf.org/html/rfc1995

4 Early Detection of Spam Domains with Passive DNS and SPF

Spam domains are sources of unsolicited mails and one of the primary vehicles for fraud and malicious activities such as phishing campaigns or malware distribution. Spam domain detection is a race: as soon as the spam mails are sent, taking down the domain or blacklisting it is of relative use, as spammers have to register a new domain for their next campaign. To prevent malicious actors from sending mails, we need to detect them as fast as possible and, ideally, even before the campaign is launched.

In this work, using near-real-time passive DNS data from Farsight Security, we monitor the DNS traffic of newly registered domains and the contents of their `TXT` records, in particular, the configuration of the Sender Policy Framework, an anti-spoofing protocol for domain names and the first line of defense against devastating Business Email Compromise scams. Because spammers and benign domains have different SPF rules and different traffic profiles, we build a new method to detect spam domains using features collected from passive DNS traffic.

Using the SPF configuration and the traffic to the `TXT` records of a domain, we accurately detect a significant proportion of spam domains with a low false positives rate demonstrating its potential in real-world deployments. Our classification scheme can detect spam domains before they send any mail, using only a single DNS query and later on, it can refine its classification by monitoring more traffic to the domain name.

This chapter is based on a work published with Maciej Korczynski and Andrzej Duda [2].

Contents

4.1 Introduction

For years, malicious mails have been representing a significant technical, economic, and social threat. Besides increasing communication costs and clogging up mailboxes, malicious mails may cause considerable harm by luring a user into following links to phishing or malware distribution sites.

Typically, malicious actors run campaigns with instant generation of a large number of mails. Hence, their detection is a race: if we want to prevent their malicious activity, we need to detect spam domain names as soon as possible, blacklist and block them (at the registration level). Once the campaign is over, domain blacklisting is less effective because the recipients have already received mails.

Early detection of spam domains that generate malicious mails is challenging. One of the approaches is to leverage the Domain Name System (DNS) that maps domain names to resource records that contain data like IP addresses. We can use DNS traffic and domain name characteristics to compute features for training and running machine learning detection algorithms, even if malicious actors may try to hide their traces and activities, and avoid domain takedown [34, 94]. The main difference between various algorithms is the set of features used to train and run classifiers. The features mainly belong to four categories: i) lexical: domain names, randomness of characters, or similarity to brand names [33, 95–100], ii) domain and IP address popularity : reputation systems based on diversity, origin of queries, or past malicious activity [33, 65, 100–104]), iii) DNS traffic: number of queries, their intensity, burst detection, or behavior changes [96, 102]), and iv) WHOIS: domain registration patterns [33, 65, 105].

In this work, we propose a scheme for early detection of spam domains, even before they send a single mail to a victim. It is based on the domain SPF (Sender Policy Framework) rules and traffic to the TXT records containing them.

SPF rules are means for detecting forged sender addresses, they form the first line of defense in the case of, for instance, Business Email Compromise scams that represented over $1.8 billion USD of losses in 2020 [106]. As malicious actors generally use newly registered domains for sending mails, they also configure the SPF rules for their domains to increase their reputation and thus avoid proactive detection. We have discovered that the content of the SPF rules and traffic to the TXT records containing them are different for malicious and benign domains. We have used these features to design a domain classifier algorithm that can quickly detect spam domains based on passive DNS traffic monitoring [39]. With low false positive rate and high true positive rate, our scheme can improve existing real-time systems for detecting and proactively blocking spam domains using passive DNS data.

The rest of this work is organized as follows. Section 4.2 provides background on SPF and spam campaigns. Section 4.3 presents the proposed scheme. Sections 4.4 and 4.5 introduce the classification algorithms and present their results. We discuss other related approaches in Section 4.6 and Section 4.7 concludes this work.

4.2 Background

In this section, we describe the SPF protocol and the mail delivery process, highlighting the steps during which we gather features to detect malicious activity.

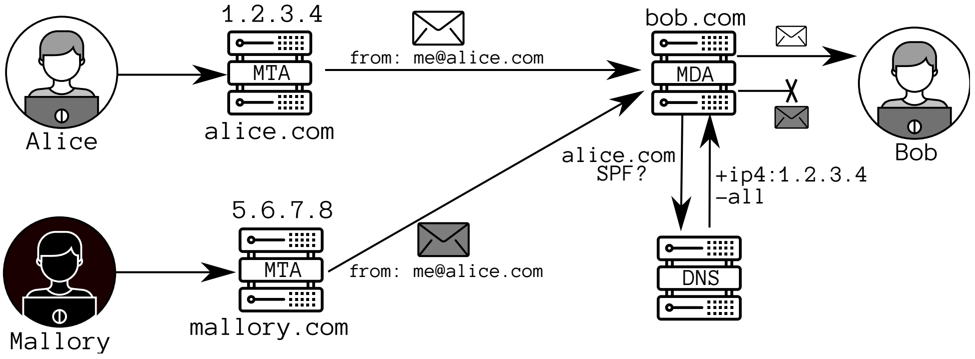


Figure 4.1: Sending mails with SPF verification.

4.2.1 Sender Policy Framework (SPF)

The Sender Policy Framework (SPF) [107] is a protocol used to prevent domain (mail) spoofing. Figure 4.1 presents the procedure for sending mails and SPF verification. Alice (sender) sends a benign mail to Bob (receiver), and Mallory (attacker) wants to send a mail that impersonates Alice to Bob. Mallory and Alice use their respective servers (`mallory.com` and `alice.com`) to send mails.

An effective anti-spoofing mechanism needs to differentiate the Mallory message from the benign Alice mail. The current first lines of defense to protect users from spoofed mails include SPF [107], DKIM [108], and DMARC [109].

SPF is a set of text-form rules in `TXT` DNS resource records specifying a list of servers allowed to send mails on behalf of a specific domain. During mail delivery over the SMTP protocol, the recipient server authenticates the sender Mail Transfer Agent (MTA) by comparing the given `MAIL FROM` (or `HELO`) identity and the sender IP address with the content of the published SPF record.

In our example, the Mail Delivery Agent (MDA) on the Bob's server queries the DNS for a `TXT` record of the sending domain (`alice.com`). This record contains the SPF rule of `alice.com` and specifies which IP addresses can send mails on behalf of this domain. The mail from Alice comes from a whitelisted server, so it gets delivered. The mail from Mallory's server was not whitelisted, so the (spoofed) mail is rejected.

A valid SPF version 1 record string must begin with `v=spf1` followed by other SPF entries with the following structure: `<qualifier><mechanism>[:<target>]`.

The mail sender is matched with the `<mechanism>:<target>` part; when matching, the output is determined by the `<qualifier>`. Four types of `<qualifier>` are possible: `PASS (+)` (the default mechanism), `NEUTRAL (~)`, `SOFTFAIL (?)`, `FAIL (-)`.

The most common SFP mechanisms are the following:

`ip4, ip6` – the sender IP address matches the predefined IP address or the subnetwork prefix,

`a, mx` – the domain has an `A` (or `MX`) record that resolves to the sender IP address,

`ptr` – a verified reverse DNS query on the sender IP address matches the sending domain (not recommended by RFC 7208 [107] since April 2014),

`exists` – the domain has an `A` record,

`include` – use the rules of another domain, `all` – the default mechanism that always matches.

To illustrate the operation of SPF rules, let us consider the following configuration for `example.com` domain: `v=spf1 a ip4:192.0.2.0/24 -all` where the `A` record (`example.com A 198.51.100.1`) is stored in DNS. The SPF rule states that only a host with the IP address of `198.51.100.1` (the `a` mechanism) or machines in the `192.0.2.0/24` subnetwork (the `ip4` mechanism) are permitted senders, all others are forbidden (the `-all` mechanism).

4.2.2 Life Cycle of a Spam Campaign

Most spam campaigns follow the same life cycle presented below.

Domain registration.

As most mail hosting companies deploy tools to prevent their users from sending spam, malicious actors need to register their own domains to send spam. To run multiple campaigns, spammers usually register domains in bulk [110]. Once the domains are registered, spammers configure zone files and fill the corresponding resource records in the DNS .

Configuration of anti-spoofing mechanisms.

To use SPF, DMARC, or DKIM, each domain must have a `TXT` resource record describing which hosts can send a mail on their behalf and deploying keys to authenticate the sender. Even if DMARC is still not widely used, many benign domains deploy SPF [111–113]. Thus, a mail from a domain without SPF configuration is likely to be flagged as spam (especially when combined with other indicators of malicious intent). To appear as benign as possible, spammers fill in at least the SPF rule in the `TXT` record. Our scheme extracts most of the features for detecting spam at this step because the SPF records of spam domains are generally different from the configurations of benign domains and even if a given domain has not yet sent a single mail, we can access its SPF rules and detect suspicious configurations. The SPF rules can be actively fetched by sending a `TXT` query to the domain (e.g., newly registered), but to avoid active scanning, we have chosen to use passive DNS to analyze `TXT` requests. In every detected spam campaign, we observe at least one `TXT` query that may originate from a spammer testing its infrastructure.

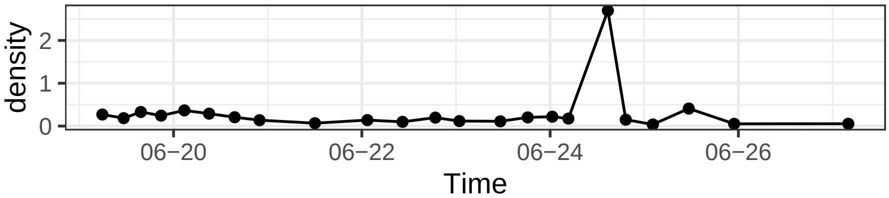


Figure 4.2: Density of DNS `TXT` traffic to a spam domain (`promotechmail.online`)

Spam campaign.

When a mail server receives a mail, it tries to resolve the `TXT` record of the sending domain to get its SPF rule and checks for possible sender forgery. During a spam campaign, spammers send mails to many servers across the world. At the beginning of a campaign, the (validating) mail servers will all try to retrieve the `TXT` DNS record of the sender domain almost at the same time. Therefore, we expect to observe a surge in queries for `TXT` records. Figure 4.2 presents traffic density (corresponding to the number of DNS queries over time, defined precisely later) to a spam domain detected during our study. The burst in the number of queries during a time window of less than 24 h, then traffic dropping and never rising again is the typical profile of spammers.

Detection, blacklisting, and cleanup.

When spam mails reach the targets, security experts and spam detection algorithms parsing the mail content and its headers flag the sending domain as a spamming source and may report it to domain blacklists like SpamHaus [114] or SURBL [115]. When a domain appears on a blacklist, mail servers will likely drop mails from it. Future spam campaigns from this domain will be unsuccessful, so it becomes useless for spammers. Hosting services may also suspend the sending server whereas domain registrars may take down the spam domain as it often violates their terms of service and is considered as DNS abuse [98, 116]. However, once the domain is blacklisted (or taken down), spammers may just acquire another one and repeat the previous steps.

When looking for spammers, timing is the key: the sooner we detect a spamming domain, the fewer mails it can send, and if an algorithm only detects a spam mail upon reception, it means that the campaign has started and reached some of the targets. This observation was the motivation for our scheme for early detection of spamming domains even before the start of a spam campaign.

4.3 Scheme for Early Detection of Spam

In this section, we present the proposed scheme. It takes advantage of passive DNS data to obtain the SPF rules for a given domain and the frequency of the queries to retrieve them.

4.3.1 Data Source: Passive DNS

Passive DNS consists of monitoring DNS traffic by *sensors* usually deployed above recursive resolvers to monitor queries between a local resolver and authoritative name servers [117]. Locally observed queries are aggregated into feeds available for analyses. In this work, we have used the near-real-time Farsight SIE Passive DNS channel 207 [39] to obtain DNS traffic data for the `TXT` records and SPF rules for each domain. We extract the following fields: the queried domain, the record type, the answer from the authoritative server, a time window, and the number of times a given query was observed during the time window.

To be effective, the scheme must analyze unencrypted DNS traffic. Therefore, it is not suitable when using the DNS over TLS (DoT) [118] or DNS over HTTPS (DoH) [119] standards that encrypt user DNS queries to prevent eavesdropping of domain names. To monitor such traffic, the scheme would have to be implemented, e.g., in public recursive resolvers providing DoT or DoH services.

4.3.2 Features Based on SPF Rules

The SPF configuration for a given domain is stored in the `TXT` record of the domain. Since most mail hosting services provide a default SPF records for their customers, many domains share the same SPF rules. Nevertheless, some domains use custom SPF rules that whitelist specific servers. We have focused on the similarities of domains: two domains that use the same custom SPF rules and whitelist the same IP addresses are likely to be managed by the same entity. Therefore, if one domain starts sending spam, it is reasonable to consider that the domains sharing the same SPF rules are likely to be (future) spammers.

We have analyzed the SPF configuration of spam and benign domains to see if they differ (we later discuss ground truth data in Section 4.4.1). Figure 4.3 shows that benign and spam domains do not necessarily use the same rules. For example, benign domains more frequently use the `+include` mechanism while spammers `+ptr`.

We presume that legitimate domains, hosted by major mail hosting providers, are more likely to have default configurations with the `+include` mechanism to indicate that a particular third party (e.g., a mail server of the provider) is authorized to send mails on behalf of all domains (e.g., in a shared hosting environment).

Spam domains may use custom mail servers instead, thus they are more likely to whitelist the IP addresses of their servers with, for instance, the `+ip4` mechanism. We suspect that in some cases spammers may not want to reveal the IP addresses of hosts sending spam. Therefore, they may use the `+all` mechanism (that accepts mails from all hosts) relatively more than legitimate domains whose administrators are concerned about rejecting spam mails from unauthorized hosts. Finally, the `+ptr` mechanism is marked as “do not use” since April 2014 by RFC 7208 [107]. Major hosting providers seem to follow this recommendation, but individual spammers may not have changed their practices and continue to use this outdated but still supported mechanism.

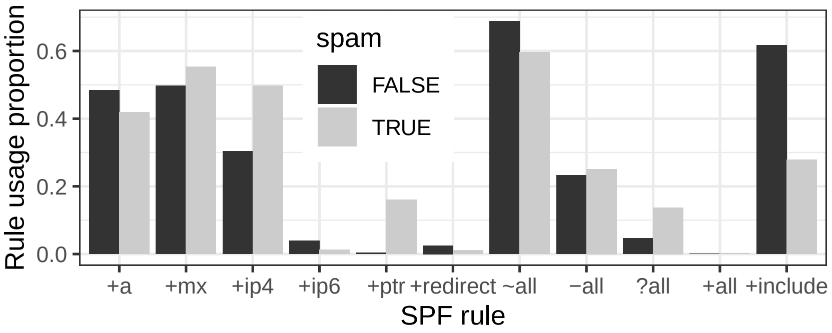
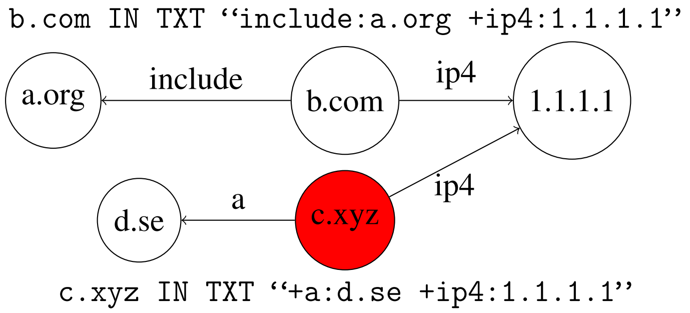


Figure 4.3: Usage proportion of SPF rules for benign and spamming domains

For each domain, we compute the number of occurrences of each mechanism in its rule to generate the set of SPF features. Because not all possible combinations of qualifiers and mechanisms are actually used, we have selected the sets of qualifiers and mechanisms that appear in more than 0.1% of domains to avoid overfitting, which leaves the ones presented in Figure 4.3.

4.3.3 Graph Analysis of SPF Rules

Some SPF rules point to an IP address or a subnetwork prefix (like `ip4` and `ip6`) and some point to domain names (like `include` and sometimes `a` and `mx`). We build the relationship graph between domains and IP ranges as shown in Figure 4.4. For example, the edge between node A (`a.org`) and node B (`b.com`) reflects the fact that node B has an SPF rule that points to node A. The edge between `b.com` and `192.0.2.1` represents the fact that this IP address is used in the `+ip4` rule in the `b.com` SPF configuration.

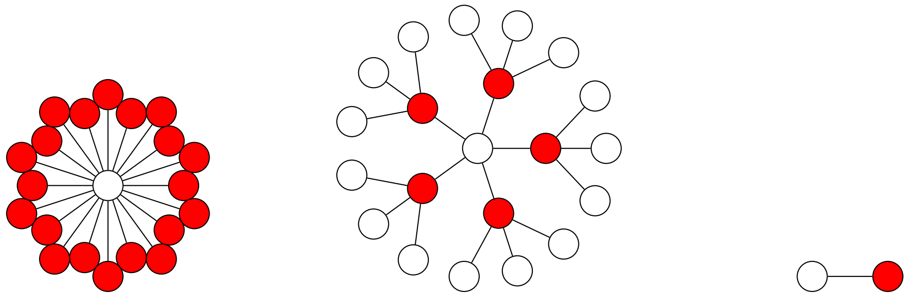


```
b.com IN TXT "include:a.org +ip4:1.1.1.1"
c.xyz IN TXT "+a:d.se +ip4:1.1.1.1"
```

Figure 4.4: Example of a relationship graph derived from SPF rules

This graph is built and updated in near real time: nodes and edges are added when domains with SPF data appear in the passive DNS feed, and spam domains (marked in red in Figure 4.4) are added or deleted from blacklists (SpamHaus and SURBL in our scheme). Thus, over time, the graph becomes more complete, providing more precise relationships and features for domain classification.

We have analyzed different structures in the graph built from our dataset and detected distinctive patterns. Figure 4.5 shows three examples of the observed structure types to illustrate some typical SPF configuration relationship graphs for spam domains. Red nodes represent spamming domains and white nodes correspond to the targets of their SPF rules. Figure 4.5a shows the pattern in which multiple spam domains share the same configuration: they have a rule targeting the same IPv6 network (these domains are likely to be managed by the same entity). Figure 4.5b presents spam domains that have an `include` mechanism that points to the same domain and exactly three other custom targets that no other domain uses (this is the case when domains are hosted by a hosting provider that provides an SPF configuration for inclusion by its clients). Finally, many spam domains have rules like in Figure 4.5c in which a domain has a single target (a custom IP address) that no other domain uses.



(a) Ball shape

(b) Tree shape

(c) One-to-One

Figure 4.5: SPF relation graph for spam domains

The study of these structures can highlight potential spam domains. In our dataset, we found structures like in Figure 4.5a or Figure 4.5b in which dozens of domains used the same rule and the majority of them appeared on spam blacklists. As such, it is reasonable to assume that the remaining domains are likely to have not yet been detected or are not yet active spam domains.

To detect the structures indicating spam domains, we have defined two unique features describing the properties of domains in the relationship graph.

Toxicity.

We define the *toxicity* of a node as the proportion of its neighbors that are flagged as spam in the graph, or 1 if the domain itself is flagged as spam. With this metric, SPF targets used by known spammers get a high value of *toxicity*. To detect the domains that use rules with high *toxicity* targets, we compute the *Max Neighbor Toxicity*: the maximum *toxicity* amongst all the targets of a domain.

This way, if a domain has a target mainly used by spammers, its *Max Neighbor Toxicity* is high.

Neighbor Degree.

For each node, we look at the degrees of its neighbors: is it connected to highly used domains and IP addresses? Or is it using custom targets that no other domain uses? We expect spamming domains to more likely use custom targets that no other domains use (with a small degree in the graph) like in Figure 4.5c, compared to benign domains that would use the default configurations of the hosting service and share the same targets as many other domains (with a high degree in the graph).

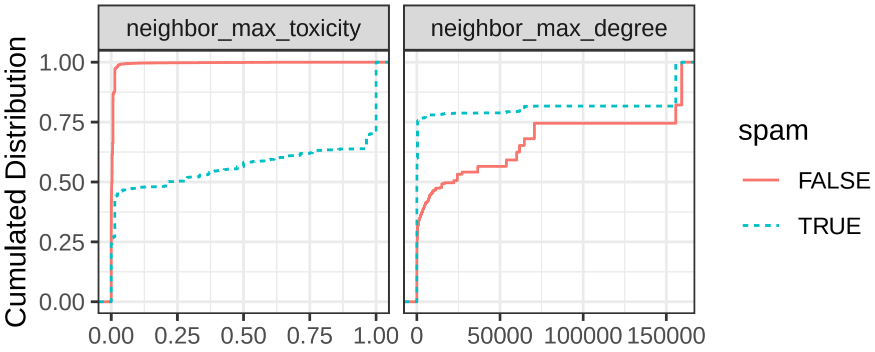


Figure 4.6: Cumulative distributions of Max Neighbor Toxicity and Max Neighbor Degree for spamming and benign domains.

Figure 4.6 shows that the expected differences of *Max Neighbor Toxicity* and *Max Neighbor Degree* between spammers and benign domains match our hypothesis: spammers are more likely to use targets shared by some other spammers and are more likely to use custom targets with low degrees in the graph.

4.3.4 Time Analysis of Traffic to DNS TXT Records

When a domain starts a spam campaign, we expect multiple servers to query DNS for the TXT record of the sender domain to check its SPF configuration. Therefore, we can observe an unusual number of queries related to the (newly registered) domain. The passive DNS feed we use contains aggregated queries over a given time window: when a DNS query is detected by a sensor, it is inserted in an aggregation buffer with the insertion timestamp. The subsequent identical queries only increase a counter in the buffer. When the buffer is full, the oldest inserted queries are flushed out, yielding an aggregated message with the query, the answer from the authoritative server, and three extra fields: `time_first`, `time_last`, and `count` meaning that the query was seen `count` times during the time window from `time_first` to `time_last`.

From these aggregated messages, we compute the traffic density by dividing the number of queries (in the `count` field) by the window duration, and then, dividing this value by the time between the end of the window and the end of the previous window to take into account the time windows in which there is no traffic. The resulting formula is the following:

$$density(i) = \frac{\texttt{count}}{\texttt{time\_last} - \texttt{time\_first}} \times \frac{1}{message\_end(i) - message\_end(i-1)}$$

Comparing the time windows of multiple domains in passive DNS data is a complex task: each window has a different size and we have no information on how the queries are spread inside it.

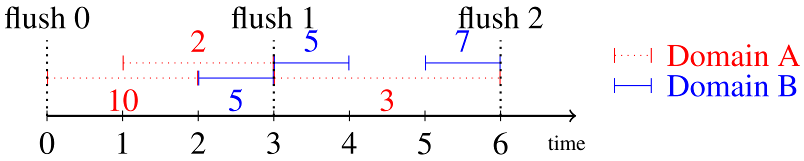


Figure 4.7: Computation of traffic density from Passive DNS messages

The query density of multiple domains can only be compared if they are computed the same way, over the same time period. If a period starts or ends in the middle of a domain time window, we need to make an assumption about how the queries are spread inside the time window, to determine how many queries are inside the time period. However, we do not have such information so to avoid unnecessary assumptions, a period can only start and end at a timestamp that it is not included in any time window. We call those usable timestamps *flushes*. Then, the query density of a domain between two flushes is computed by measuring the time during which the domain was active, the total time between the flushes and the number of queries. For example, in Figure 4.7, between flush 0 and 1, Domain A has a `count` (total number of queries) of 12 and an `active_time` (total time covered by time windows) of 3, and Domain B has a `count` of 5, and an `active_time` of 1. If $flush(i)$ is the timestamp of the i-th flush, we define the density at time i as:

$$density(i) = \frac{count}{active\_time} \times \frac{1}{flush(i+1) - flush(i)}.$$

The first fraction represents the density of requests in the aggregated time window. The second fraction normalizes this value by the size of the flush window so that all domains have a comparable density, as the flushes are not evenly spread. Therefore, $density(0)$ for domain A is $12/3 \times 1/3 = 4/3$ and $5/1 \times 1/3 = 5/3$ for domain B .

Max Variation.

To detect large variations in density, we compute the *Max Variation* feature defined as the maximum density variation during 24 h. Domains with a slowly increasing traffic have a low *Max Variation* and those with a spike in the number of TXT queries, a high *Max Variation*. We compute two versions of this feature: i) the *Global Max Variation*, using the same time steps to compare all domains and ii) the *Local Max Variation* in which a custom time step is computed for each domain.

For the *Max Global Variation*, the *flushes* are computed using the time windows of all the studied domains at the same time (the numbered flushes in Figure 4.7), meaning that a timestamp is a *flush* only if no domain has a window opened at this time. This results in fewer *flushes* but the traffic density between different domains can be compared (as they all use the same time steps). The *Max Local Variation* of a domain is computed using only the time windows of this domain to compute the *flushes* (numbered *flushes* plus domain *flushes* in Figure 4.7). The *Max Local Variation* uses more time steps so the density is more precise, but these time steps are different for each domain and have a tendency to reduce the detection of sudden bursts following a long inactivity window.

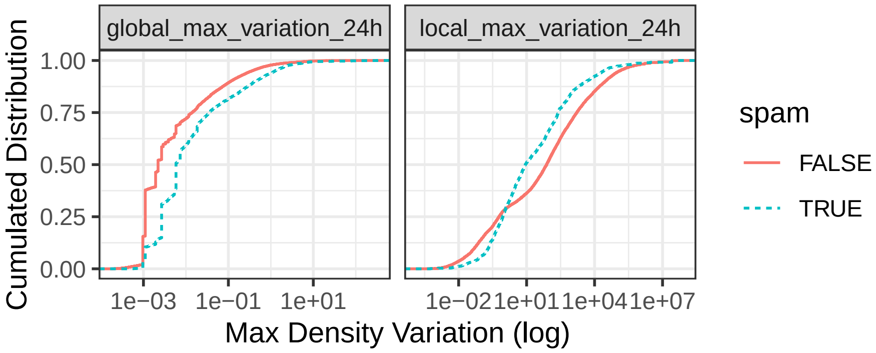


Figure 4.8: Cumulative distribution of Max Variation (log scale x-axis)

Figure 4.8 presents the cumulative distribution of the two features. As expected, we observe that spam domains have a relatively higher *Max Global Variation* when all domains share the same time steps.

However, when we look at the *Max Local Variation*, we observe that benign domains tend to have a higher variation. The distributions are different because this feature is close to the average density variation: domains with a lot of traffic variation and small windows will have a higher *Local Variation*, whereas spam domains with almost no traffic except for a few spikes will have a lower *Local Variation* due to long periods of inactivity before a spike.

4.4 Classifiers

In this section, we present the classifiers used for the detection of spam based on the proposed features.

4.4.1 Ground Truth

We have taken the precaution of carefully selecting the domains in our ground truth. We recorded four months (between May and August 2021) of passive DNS traffic to TXT records from Farsight Security [39]. Because most spam domains are newly registered and discarded as soon as they are blacklisted, we only considered newly registered domains. From the ICANN Central Zone Data Service (CZDS) [42], we have built a list of new domains by computing the difference between consecutive versions of each generic Top Level Domain (gTLD) zone files.

Table 4.1 shows the number of queries and unique domains at each data collection and analysis stage. The first step captures DNS TXT queries to newly registered domain names observed in the passive DNS feed. The next step retains only the TXT queries that contain valid SPF data. Then, we build ground truth with the approach described in Section 4.4.1.

Table 4.1: Number of queries and unique domains in the dataset at different stages

Stage	Queries	Unique domains	Spam domains
1. Traffic to new domains	399M	14M	0.8%
2. SPF traffic	36M	1.4M	1.5%
3. Ground truth	26M	40,224	5.9%

Using SURBL [115] and SpamHaus [114] spam blacklists, we have identified all domains (in near-real time) in our database flagged by one of these sources. Spam blacklists are not perfect and sometimes they may flag benign domains as spam. Therefore, to obtain reliable ground truth, we added an extra layer of verification: a domain is labeled as

- **benign** if it has not been blacklisted and has been active during the entire period of the study (and has a valid A and NS records), or
- **malicious** if it was blacklisted by SURLB or SpamHaus and was taken down.

With these criteria, our ground truth dataset contained 37,832 non-spam and 2,392 spam domains.

4.4.2 Classifier

For spam detection, it is crucial to keep the True Negative[1] Rate (TPR) as high as possible to avoid flagging benign domains as spam, because wrongly classifying a benign domain as spam can have serious repercussions, like domain takedown or blacklisting. Once a True Negative Rate of at least 99% is achieved, we maximize the True Positive[2] Rate (TPR) to detect as many spam domains as possible.

The performance of each classifier is measured with three metrics:

[1] True Negative: non-spam domain correctly classified as such

[2] True Positive: spam domain correctly classified as malicious

F1-score: $\frac{2TP}{2TP+FP+FN}$, with TP, FP and FN being respectively the number of True Positives, False Positives, False Negatives

True Positive Rate (TPR): $\frac{TP}{TP+FN}$: proportion of spam domains accurately flagged as spam.

True Negative Rate (TNR): $\frac{TN}{TN+FP}$: proportion of benign domains accurately flagged as benign.

To calculate performance metrics, we use the k-fold technique: the whole ground truth dataset is split in 5 equal parts called folds. We select one fold for testing the model performance and train the model using the $k-1$ remaining folds. We repeat this process for each fold (each fold is evaluated against the remaining $k-1$). Once the model for each fold is built, the global efficiency of the model is the average efficiency of the five iterations.

We explored multiple classifiers and parameters with Weka [120], then implemented two of them with the `scikit-learn` [121] Python library, for better benchmarking. Two classifiers that performed the best are:

C4.5 or J48: a decision tree able to describe non-linear relations between features. It highlights complex conditional relations between features.

Random Forest: a set of multiple decision trees with a voting system to combine their results. Its drawback is low explainability.

Table 4.2: Features used by the classifiers

Category	Feature	Outcome
SPF Rules	Number of…	
	`+all, +mx, +ptr, -all`	Malicious
	`+a, +include, +redirect, ~all`	Benign
	`+ip4, +ip6, ?all`	Mixed[3]
SPF Graph	Max Neighbor Degree	Benign
	Max Neighbor Toxicity	Malicious
Time Analysis	Max Global Variation	Malicious
	Max Local Variation	Benign

We use the k-fold cross-validation technique with k set to 5, as described in Section 4.4.2.

The number of spam domains in our ground truth dataset represents less than 10% of all domains. The decision tree algorithms are not suitable for classification problems with a skewed

[3] Depends on how many times the rule is present in the configuration

class distribution. Therefore, we have used a standard class weight algorithm for processing imbalanced data [122] implemented in the `scikit-learn` Python library [121].

Table 4.2 summarizes the features used by the classifiers and whether they indicate maliciousness or benignness of the domain.

4.5 Classification Results

We evaluate the efficiency of the classifiers with two sets of features: i) the *static* set without the time analysis features (Max Variation) and ii) the *static + dynamic* set that includes both static and the time analysis features. We have distinguished between the sets because even if the efficiency is lower without the time analysis features, we can get the static features (SPF configuration and graph properties) from a single `TXT` query to the target domain allowing for a rapid detection of most spam domains. Then, we can refine the classification by adding the time based features that are more robust against evasion techniques but require more time to detect spam domains.

4.5.1 Performance Evaluation

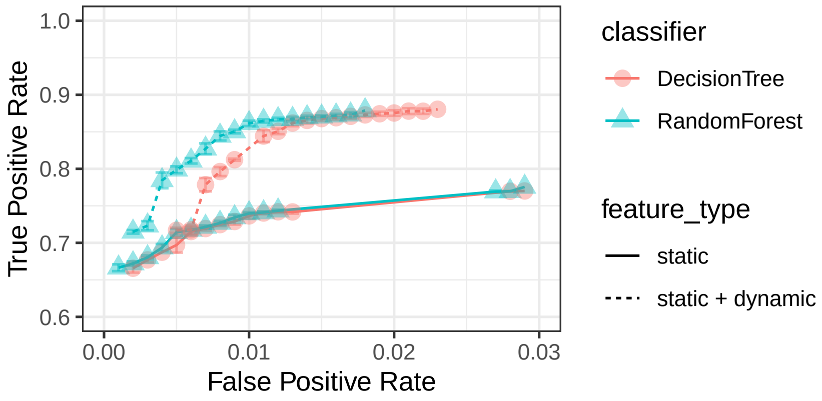


Figure 4.9: ROC curve for different classifiers on two sets of features

Figure 4.9 compares the Receiver Operating Characteristic (ROC) curves of each classifier for two sets of features (to see better the differences in performance, we zoom into high values of TPR). When training the classifiers, we change the weight of the spam class to change the reward of accurately finding a spam domain. If the spam class weight is low, the classifier will be less likely to risk getting a false positive. On the contrary, if the spam class weight is high, the classifier gets higher reward if it accurately flags a spam domain. Therefore, the classifier will "take more risks", reducing its TNR to increase TPR. If we require the False Positive Rate (benign domains flagged as spam) under 1%, the Random Forest is the best algorithm reaching a True Positive Rate of 74% using only the static set and 85% once we add the time analysis features.

Table 4.3 shows the results of the Random Forest classifier using static and dynamic features (SPF Rules, SPF Graph and Time Analysis features). It corresponds to the model from Figure 4.9 with a TPR of 0.717 and FPR of 0.006. The second and third columns (Spam and Benign) represent how commercial blacklists (SpamHaus and SURBL) classified the domains (ground truth data), whereas the second and third row represent how our system classified the

Table 4.3: Classification results for the Random Forest classifier on the ground truth dataset.

Blacklists Our method	Spam	Benign	**Total**
Spam	TP = 1 716	FP = 210	**1 926**
Benign	FN = 676	TN = 37 622	**38 298**
Total	**2 392**	**37 832**	**40 224**
	TPR	**TNR**	**F1-score**
	71.7%	99.4%	79.5%

same domains. The second part of the table represents common metrics used to rate classifiers. For example, in the table we can note that 676 domains were classified as Benign by our classifier, but they appear in the commercial blacklists—this represents the number of False Negatives (FN). The second part of the table shows the metrics used to evaluate our classifier (TPR, TNR, and F1-score) as described in Section 4.4.2.

Figure 4.10 illustrates how long we need to monitor a domain so that the classifiers reach their best efficiency. Over time, we observe traffic to each domain and the time analysis features get more precise (until one week), which improves classification. Both classifiers reach almost the best detection performance (computed as the F1-score) after observing a domain for one day.

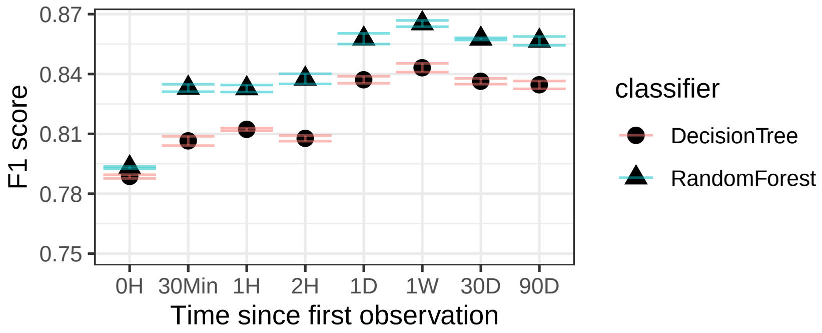


Figure 4.10: F1-score of classifiers after the first appearance of each domain

4.5.2 Detection Time

The static results (labeled as 0H in Figure 4.10) show the efficiency of the scheme when a single TXT request is observed. In this case, the classifier has no time properties of the traffic and only uses the static features (SPF Rules and SPF Graph). We can replace passive detection of SPF Rules with active DNS scans (assuming we have a list of newly registered domain names, which is generally the case for legacy and new gTLDs but not for the vast majority of ccTLD [116, 123]): by actively querying the TXT records of new domains and classifying them based on their

SPF configuration and formed relationships. Then, over time, as we passively observe traffic to the domain records, the performance of the classifier improves achieving very good results after 30 minutes (F1-score of 0.83) of monitoring (with Random Forest) in comparison with the F1-score of 0.86 after one day.

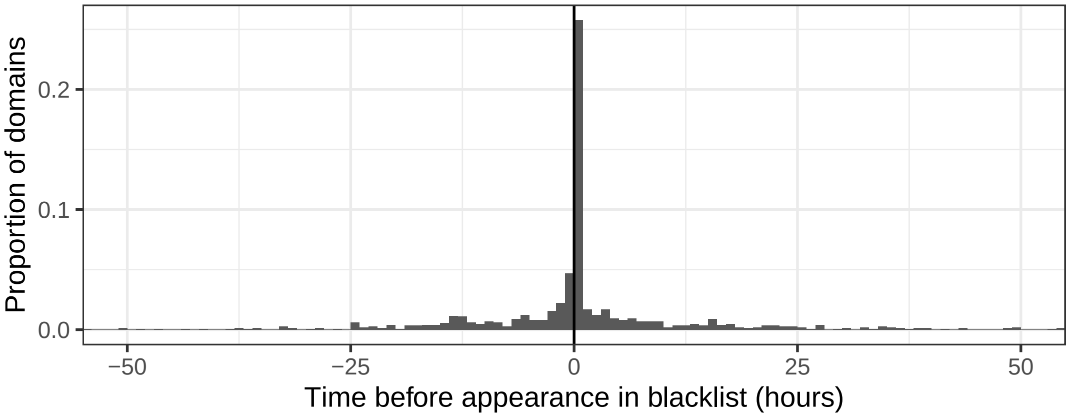


Figure 4.11: Time before detected spam domains appear in commercial blacklists

Using only static features, we compared the spam domain detection speed of our scheme with two commercial blacklists (SpamHaus and SURBL). In Figure 4.11, we plotted the time elapsed between the detection by our scheme and the appearance of domains in the blacklists (with an hourly granularity). We limited the graph to 50 hours, but considerable number of domains only appear in the commercial blacklists weeks after we detect them. Positive values mean that our scheme was faster: for 70% of the detected spam domains, our scheme was faster than the commercial blacklists. However, 26% of the domains detected by our scheme appear in the commercial blacklists in the following hour, whereas 30% of the domains are detected more than 24 hours in advance. The negative values represent domain names where our scheme was slower than the commercial blacklists: 30% of the domains were already in the blacklists when they were observed in our passive DNS feed for the first time and classified as spam.

4.5.3 Feature Importance

The importance of each feature was computed by looking at how selective the feature was in the Random Forest classifier [121]. The importance of each feature and each category is described in Table 4.4. It is not a surprise that the Maximum Neighbor Toxicity is by far the most important feature: a domain whitelisting the same IP addresses and domains as a known spamming domain is very likely to be managed by spammers. The most important SPF rule for classification is `+ptr`: as we discussed in Section 4.3.2, this rule is almost never used by benign domains (following the RFC 7208 recommendations). Lastly, the Global Max Variation is the most important dynamic feature: massive increases in the number of queries to a domain is a distinctive trait of spamming domains, as presented in Section 4.2.2, but this feature is only useful after the start of the spam campaign.

Table 4.4: Importance of each feature for the Random Forest classifier

Feature	Importance
SPF Graph features	**0.574515**
`neighbor_max_toxicity`	0.463689
`neighbor_max_degree`	0.110826
SPF Rules features	**0.232846**
`+ptr`	0.100481
`+a`	0.029005
`+ip4`	0.028789
`+mx`	0.021006
`+include`	0.017561
`?all`	0.013728
`~all`	0.011522
Other rules	< 0.01
Time Analysis features	**0.192638**
`global_max_variation_24h`	0.122167
`local_max_variation_24h`	0.036828
`global_max_triggers_24h`	0.022380
`local_max_triggers_24h`	0.011263

4.6 Related Work

The four main categories of features used to detect malicious domains are the following: i) Lexical: domain name, randomness of characters, or similarity to brand names [33,96,99,100,105], ii) Domain and IP address popularity: reputation systems based on diversity, origin of queries, or past malicious activity [33, 65, 100, 103–105], iii) DNS traffic: number of queries, intensity, burst detection, behavior changes [96, 102], iv) WHOIS (domain registration data): who registered a given domain[4], when, and at which registrar [33, 65, 105]. Other methods develop specific features extracted from the content of mails: size of the mail, links, or redirections [105,125]. With the selected features, machine learning algorithms classify malicious and benign domains.

With respect to the methods that work on passive data such as Exposure [96] that need some time to detect abnormal or malicious patterns, we focus on early detection of spam domains. Exposure for instance, needs around a week of observation before possible detection, while we achieve a F1-score of 79% based on a single DNS query. Our scheme can be applied at early stages of a domain life cycle: using passive (or active) DNS, we can obtain SPF rules for newly registered domains and classify them immediately, or wait until we detect TXT queries to that domain and refine the classification using hard-to-evade temporal features.

Other methods generally try to detect abnormal or malicious patterns at later phases of the domain life cycle. Schemes based on content or long period traffic analysis may reach high efficiency but generally cannot run before or at the beginning of an attack. Schemes using lexical and popularity features can run preemptively but may have reduced efficiency, compared to dynamic schemes.

Our scheme may complement other approaches that aim at detecting spam during other phases in the life cycle of spam campaigns and other algorithms that rely on a variety of different features.

4.7 Conclusion

In this work, we have proposed a new scheme for early detection of spam domains based on the content of domain SPF rules and traffic to the TXT records containing them. With this set of features, our best classifier detects 85% of spam domains while keeping a False Positive Rate under 1%. The detection results are remarkable given that the classification only uses the content of the domain SPF rules and their relationships, and hard to evade features based on DNS traffic. The performance of the classifiers stays high, even if they are only given the static features that can be gathered from a single TXT query (observed passively or actively queried).

With a single request to the TXT record, we detect 75% of the spam domains, possibly before the start of the spam campaign. Thus, our scheme brings important speed of reaction: we can detect spammers with good performance even before any mail is sent and before a spike in the DNS traffic. To evaluate the efficiency of the proposed approach based on passive DNS, we did not combine the proposed features with other ones used in previous work like domain registration patterns [33, 65, 105]. In practical deployments, the classification can be improved by adding other features based on, e.g., the content of potentially malicious mails or the lexical patterns of the domain names.

[4] not available after the introduction of the General Data Protection Regulation (GDPR) and the ICANN Temporary Specification [124].

The features used in our scheme yield promising results, so adding them to existing spam detection systems will increase their performance without large computation overhead as SPF data can easily be extracted from near-real -time passive DNS feeds already used in some schemes.

Acknowledgements

We thank Paul Vixie and Joe St Sauver (Farsight Security), the reviewers, our shepherd, and Sourena Maroofi for their valuable and constructive feedback. We thank Farsight Security for providing access to the passive DNS traffic as well as SpamHaus and SURBL for the spam blacklists. This work was partially supported by the Grenoble Alpes Cybersecurity Institute under contract ANR-15-IDEX-02 and by the DiNS project under contract ANR-19-CE25-0009-01.

Conclusion and perspectives

This thesis explored many aspects of the Domain Name System: its architecture, its traditional uses, how it can support other protocols and how it can be used to detect malicious activities. In the first chapter, after providing a general description of the DNS, we presented how this system can be used by users and protocols, abused by malicious actors and studied by researchers and experts. Then we described the three main contributions of this work, in the form of published articles and works in progress. Finally, in this chapter, we recapitulate the contributions of this thesis and discuss additional works in progress and future research perspectives.

4.8 Contributions

In the first chapter of this work, we provided a generic introduction to DNS, its protocol and the architecture effectively implementing this system. Through multiple real-life examples we highlighted how this protocol is a crucial piece of the modern internet and how it is often the first step of many other systems and algorithms by associating IP addresses to domain names. We described how the DNS is still being abused as of today, both as a direct target of malicious attacks and as a way to support, deploy and hide other malicious architectures. This being said, we also described how the DNS protocol and architecture evolved over the years, with multiple protocol extensions to update this protocol as old as the Internet and adapt it to the threats and challenges of the modern Internet.

In the second chapter of this work, we examined the core assumption that WHOIS and RDAP databases, holding registration information of domain names, offer the same data and that users can query them interchangeably. By collecting, processing, and comparing 164 million entries for a sample of 55 million domain names, we revealed that while the data obtained through WHOIS and RDAP is generally consistent, 7.6% of the observed domains still present inconsistent data on critical fields like nameservers, IANA ID, or creation date. This raises concerns about the trust placed by the community on the coherence of these databases. We will contribute to the community all the data gathered during this study, opening the way for more in-depth analysis on specific inconsistencies.

In the third chapter, we focused on the Internet of Things and constrained devices. We developed naming schemes to encode device properties like sensor type or geographical location. We adapted this naming scheme to the constraints of the DNS to leverage this protocol and provide an efficient distributed way to discover devices based on their properties. To prove the feasibility of our design, we worked in collaboration with the University of Parma to build two prototypes using this naming scheme to discover constrained LoRa devices. This work highlights how the DNS can be used for service discovery, even for devices that do not use the IP protocol like LoRa devices, and describes how it can be used to share sensor data while leveraging the powerful decentralization and cache of the DNS architecture.

In the fourth chapter of this work, we used the DNS as a way to detect spam domains before the start of mailing campaigns. Using near-real-time passive DNS data, we monitored the DNS traffic of newly registered domains and the contents of their TXT records, in particular, the configuration of the Sender Policy Framework, an anti-spoofing protocol for domain names.

Because spammers and benign domains have different SPF rules and different traffic profiles, we built a new method to detect spam domains using features collected from passive DNS traffic. Using the SPF configuration and the traffic to the TXT records of a domain, we accurately detected a significant proportion of spam domains with a low false positives rate demonstrating its potential in real-world deployments. Our classification scheme can detect spam domains before they send any mail, using only a single DNS query and later on, it can refine its classification by monitoring more traffic to the domain name.

4.9 Future Work

We are currently working with administrators of the B-Root DNS servers to study the deployment of recursive-to -authoritative DNS encryption. The encryption of these communications are often overlooked as user privacy protection mechanisms already exist and personal data is harder to extract from messages between resolvers and authoritative servers, while DNSSEC can prevent packet modification and Man-in-the-Middle attacks. However, those mechanisms are imperfect and still leave challenges that could be addressed by resolver-to -authoritative encryption. With this work in progress, we are studying the potential costs and consequences of this encryption for resolvers and authoritative servers, to help server administrators make educated choices on this topic and provide solid bases for the community to dive deeper on this topic.

Working with Passive DNS data, we observed several unexpected behaviors, like queries to domains that should always stay inside their origin network or proofs of misconfigurations that were revealed decades ago and that are still not fixed. A deeper analysis on this unexpected Passive DNS traffic could shed light on many malicious behaviors, misconfigurations and vulnerabilities currently present on the network.

Finally, we would also like to use DNS servers as a reference point in Internet networks. The network position of authoritative servers, and more specifically root servers, rarely change, therefore the routes between them or other stable points on the Internet can be considered as relatively stable. Studying the modifications of these routes could help detect wide-scale network changes and attacks towards the BGP routing protocol.

Bibliography

[1] S. Fernandez, M. Amoretti, F. Restori, M. Korczynski, and A. Duda, “Semantic Identifiers and DNS Names for IoT,” in *2021 International Conference on Computer Communications and Networks (ICCCN)*, (Athens, Greece), pp. 1–9, IEEE, July 2021.

[2] S. Fernandez, M. Korczyński, and A. Duda, “Early Detection of Spam Domains with Passive DNS and SPF,” in *Passive and Active Measurement* (O. Hohlfeld, G. Moura, and C. Pelsser, eds.), vol. 13210, pp. 30–49, Cham: Springer International Publishing, 2022.

[3] J. Bayer, Y. Nosyk, O. Hureau, S. Fernandez, I. Paulovics, A. Duda, and M. Korczyński, *Study on Domain Name System (DNS) Abuse: Technical Report*. 2022.

[4] Y. Nosyk, O. Hureau, S. Fernandez, A. Duda, and M. Korczyński, “Unveiling the Weak Links: Exploring DNS Infrastructure Vulnerabilities and Fortifying Defenses,” in *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, (Delft, Netherlands), pp. 546–557, IEEE, July 2023.

[5] “Host names on-line,” Request for Comments RFC 608, Internet Engineering Task Force, Jan. 1974.

[6] Mockapetris, “Domain names - concepts and facilities,” Request for Comments RFC 1034, Internet Engineering Task Force, Nov. 1987.

[7] IETF, “Domain names - implementation and specification,” Request for Comments RFC 1035, Internet Engineering Task Force, Nov. 1987.

[8] L. Daigle, “WHOIS Protocol Specification,” Request for Comments RFC 3912, Internet Engineering Task Force, Sept. 2004.

[9] S. Hollenbeck and A. Newton, “Registration Data Access Protocol (RDAP) Query Format,” Request for Comments RFC 9082, Internet Engineering Task Force, June 2021.

[10] E. Union, “General data protection regulation,” 2016. https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[11] B. Wellington, “Secure Domain Name System (DNS) Dynamic Update,” Request for Comments RFC 3007, Internet Engineering Task Force, Nov. 2000.

[12] B. Wellington, “Domain Name System Security (DNSSEC) Signing Authority,” Request for Comments RFC 3008, Internet Engineering Task Force, Nov. 2000.

[13] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. E. Hoffman, “Specification for DNS over Transport Layer Security (TLS),” Request for Comments RFC 7858, Internet Engineering Task Force, May 2016.

[14] P. E. Hoffman and P. McManus, "DNS queries over HTTPS (DoH)." RFC 8484, 2018.

[15] C. Huitema, S. Dickinson, and A. Mankin, "DNS over dedicated QUIC connections." RFC 9250, 2022.

[16] J. C. Klensin, "Simple Mail Transfer Protocol," Request for Comments RFC 5321, Internet Engineering Task Force, Oct. 2008.

[17] S. Kitterman, "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1," Request for Comments RFC 7208, Internet Engineering Task Force, Apr. 2014.

[18] D. Crocker, T. Hansen, and M. Kucherawy, "RFC 6376: DomainKeys Identified Mail (DKIM) Signatures." Internet Requests for Comments, 2011.

[19] E. Kucherawy, M. Zwicky, and E. Zwicky, "RFC 7489: Domain-Based Message Authentication, Reporting, and Conformance (DMARC)." Internet Requests for Comments, 2015.

[20] D. N. I. B. Verisign, "The Domain Name Industry Brief Quarterly Report."

[21] IANA, "Root Zone Database." https://www.iana.org/domains/root/db.

[22] "H.root-servers.net Statistics." https://h.root-servers.org/.

[23] IANA, "List of tlds," 2023. https://www.iana.org/domains/root/db.

[24] H. Nielsen, R. T. Fielding, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.0," Request for Comments RFC 1945, Internet Engineering Task Force, May 1996.

[25] C. M. Lonvick and T. Ylonen, "The Secure Shell (SSH) Authentication Protocol," Request for Comments RFC 4252, Internet Engineering Task Force, Jan. 2006.

[26] "File Transfer Protocol," Request for Comments RFC 114, Internet Engineering Task Force, Apr. 1971.

[27] P. Resnick, "Internet Message Format," Request for Comments RFC 5322, Internet Engineering Task Force, Oct. 2008.

[28] ICANN, "Factsheet - DNS attack 6 February 2007."

[29] G. C. Moura, R. D. O. Schmidt, J. Heidemann, W. B. De Vries, M. Muller, L. Wei, and C. Hesselman, "Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event," in *Proceedings of the 2016 Internet Measurement Conference*, (Santa Monica California USA), pp. 255–270, ACM, Nov. 2016.

[30] S. Son and V. Shmatikov, "The hitchhiker's guide to DNS cache poisoning," in *Security and Privacy in Communication Networks: 6th International ICST Conference, SecureComm 2010, Singapore, September 7-9, 2010. Proceedings 6*, pp. 466–483, Springer, 2010.

[31] M. Korczyński, M. Król, and M. van Eeten, "Zone Poisoning: The How and Where of Non-Secure DNS Dynamic Updates," in *Proc. of IMC*, p. 271–278, 2016.

[32] T. Palmer, "First try DNS cache poisoning with IPv4 and IPv6."

[33] V. Le Pochat, T. Van hamme, S. Maroofi, T. Van Goethem, D. Preuveneers, A. Duda, W. Joosen, and M. Korczynski, "A Practical Approach for Taking Down Avalanche Botnets Under Real-World Constraints," in *Proceedings 2020 Network and Distributed System Security Symposium*, (San Diego, CA), Internet Society, 2020.

[34] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and Detecting Fast-Flux Service Networks," p. 12.

[35] M. Wander, "Measurement survey of server-side DNSSEC adoption," in *2017 Network Traffic Measurement and Analysis Conference (TMA)*, (Dublin, Ireland), pp. 1–9, IEEE, June 2017.

[36] C. Rossow, "Amplification Hell: Revisiting Network Protocols for DDoS Abuse," in *In Proc. of NDSS*, 2014.

[37] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting Algorithmically Generated Domain-Flux Attacks With DNS Traffic Analysis," *IEEE/ACM Transactions on Networking*, vol. 20, pp. 1663–1677, Oct. 2012.

[38] R. Perdisci, I. Corona, and G. Giacinto, "Early Detection of Malicious Flux Networks via Large-Scale Passive DNS Traffic Analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, pp. 714–726, Sept. 2012.

[39] F. Inc., "Farsight SIE."

[40] S. Europe, "SIE Europe."

[41] T. Grzinic, D. Perhoc, M. Maric, F. Vlasic, and T. Kulcsar, "CROFlux-2014; Passive DNS method for detecting fast-flux domains," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, (Opatija, Croatia), pp. 1376–1380, IEEE, May 2014.

[42] ICANN, "ICANN: Centralized Zone Data Service." `https://czds.icann.org`.

[43] E. P. Lewis and A. Hoenes, "DNS Zone Transfer Protocol (AXFR)," Request for Comments RFC 5936, Internet Engineering Task Force, June 2010.

[44] L. Izhikevich, G. Akiwate, B. Berger, S. Drakontaidis, A. Ascheman, P. Pearce, D. Adrian, and Z. Durumeric, "ZDNS: A fast DNS toolkit for internet measurement," in *Proceedings of the 22nd ACM Internet Measurement Conference*, (Nice France), pp. 33–43, ACM, Oct. 2022.

[45] ICANN, "Icann temporary agreement for gtlds to comply with gdpr," 2023. `https://www.icann.org/resources/pages/gtld-registration-data-specs-en`.

[46] S. Hollenbeck and A. Newton, "JSON Responses for the Registration Data Access Protocol (RDAP)," Request for Comments RFC 9083, Internet Engineering Task Force, June 2021.

[47] T. Lauinger, K. Onarlioglu, A. Chaabane, W. Robertson, and E. Kirda, “WHOIS Lost in Translation: (Mis)Understanding Domain Name Expiration and Re-Registration,” in *Proceedings of the 2016 Internet Measurement Conference*, (Santa Monica California USA), pp. 247–253, ACM, Nov. 2016.

[48] C. Ganan, “WHOIS sunset? A primer in Registration Data Access Protocol (RDAP) performance,” *TMA*, p. 8, 2021.

[49] C. Lu, B. Liu, Y. Zhang, Z. Li, F. Zhang, H. Duan, Y. Liu, J. Q. Chen, J. Liang, Z. Zhang, S. Hao, and M. Yang, “From WHOIS to WHOWAS: A Large-Scale Measurement Study of Domain Registration Privacy under the GDPR,” in *Proceedings 2021 Network and Distributed System Security Symposium*, (Virtual), Internet Society, 2021.

[50] M. Maass, A. Stöver, H. Pridöhl, S. Bretthauer, D. Herrmann, M. Hollick, and I. Spiecker, “Effective Notification Campaigns on the Web: A Matter of Trust, Framing, and Support,” *USENIX Security 2021*, 2021.

[51] ICANN, “Icann registrar agreement,” 2012. https://www.icann.org/resources/pages/registrars-0d-2012-02-25-en.

[52] J. Gould, “Extensible Provisioning Protocol (EPP) and Registration Data Access Protocol (RDAP) Status Mapping,” Request for Comments RFC 8056, Internet Engineering Task Force, Jan. 2017.

[53] ICANN, “Icann whois history.” https://whois.icann.org/en/history-whois.

[54] A. Newton and S. Hollenbeck, “Registration Data Access Protocol (RDAP) Query Format,” Request for Comments RFC 7482, Internet Engineering Task Force, Mar. 2015.

[55] A. Newton and S. Hollenbeck, “JSON Responses for the Registration Data Access Protocol (RDAP),” Request for Comments RFC 7483, Internet Engineering Task Force, Mar. 2015.

[56] M. Blanchet, “Finding the Authoritative Registration Data Access Protocol (RDAP) Service,” Request for Comments RFC 9224, Internet Engineering Task Force, Mar. 2022.

[57] M. Loffredo and M. Martinelli, “Registration Data Access Protocol (RDAP) Partial Response,” Request for Comments RFC 8982, Internet Engineering Task Force, Feb. 2021.

[58] S. Hollenbeck and A. Newton, “Registration data access protocol (rdap) object tagging,” Request for Comments RFC 8521, Internet Engineering Task Force, Nov. 2018.

[59] F. A. Ghaleb, M. Alsaedi, F. Saeed, J. Ahmad, and M. Alasli, “Cyber Threat Intelligence-Based Malicious URL Detection Model Using Ensemble Learning,” *Sensors*, vol. 22, p. 3373, Apr. 2022.

[60] T. Vissers, W. Joosen, and N. Nikiforakis, “Parking Sensors: Analyzing and Detecting Parked Domains,” in *Proceedings 2015 Network and Distributed System Security Symposium*, (San Diego, CA), Internet Society, 2015.

[61] S. Liu, I. Foster, S. Savage, G. M. Voelker, and L. K. Saul, “Who is .com?: Learning to Parse WHOIS Records,” in *Proceedings of the 2015 Internet Measurement Conference*, (Tokyo Japan), pp. 369–380, ACM, Oct. 2015.

[62] A. McCallum and W. Li, “Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons,” in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 -*, vol. 4, (Edmonton, Canada), pp. 188–191, Association for Computational Linguistics, 2003.

[63] N. Christin, S. S. Yanagihara, and K. Kamataki, “Dissecting one click frauds,” in *Proceedings of the 17th ACM Conference on Computer and Communications Security - CCS ’10*, (Chicago, Illinois, USA), p. 15, ACM Press, 2010.

[64] M. Felegyhazi, C. Kreibich, and V. Paxson, “On the Potential of Proactive Domain Blacklisting,” *LEET 2010*, 2010.

[65] S. Hao, A. Kantchelian, B. Miller, V. Paxson, and N. Feamster, “PREDATOR: Proactive Recognition and Elimination of Domain Abuse at Time-Of-Registration,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, (Vienna Austria), pp. 1568–1579, ACM, Oct. 2016.

[66] A. Affinito, R. Sommese, G. Akiwate, S. Savage, KC. Claffy, G. M. Voelker, A. Botta, and M. Jonker, “Domain Name Lifetimes: Baseline and Threats,” *TMA 2022*, p. 9, 2022.

[67] V. Le Pochat, T. Van hamme, S. Maroofi, T. Van Goethem, D. Preuveneers, A. Duda, W. Joosen, and M. Korczynski, “A Practical Approach for Taking Down Avalanche Botnets Under Real-World Constraints,” in *Proceedings 2020 Network and Distributed System Security Symposium*, (San Diego, CA), Internet Society, 2020.

[68] K. Du, H. Yang, and Z. Li, “The Ever-changing Labyrinth: A Large-scale Analysis of Wildcard DNS Powered Blackhat SEO,” *USENIX Security 2016*, p. 19, 2016.

[69] O. Çetin, M. Hanif Jhaveri, C. Gañán, M. van Eeten, and T. Moore, “Understanding the role of sender reputation in abuse reporting and cleanup,” *Journal of Cybersecurity*, vol. 2, pp. 83–98, Dec. 2016.

[70] IANA, “Registrar ids,” 2023. https://www.iana.org/assignments/registrar-ids/registrar-ids.xhtml.

[71] C. Partridge and M. Allman, “Ethical Considerations in Network Measurement Papers,” *Commun. ACM*, vol. 59, p. 58–64, sep 2016.

[72] D. Dittrich and E. Kenneally, “The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research,” 2012.

[73] R. Sommese, G. C. M. Moura, M. Jonker, R. van Rijswijk-Deij, A. Dainotti, K. C. Claffy, and A. Sperotto, “When Parents and Children Disagree: Diving into DNS Delegation Inconsistency,” in *Passive and Active Measurement* (A. Sperotto, A. Dainotti, and B. Stiller, eds.), vol. 12048, pp. 175–189, Cham: Springer International Publishing, 2020.

[74] A. P. Bianzino, D. Pezzuolo, and G. Mazzini, “Who is whois? An analysis of results consistence,” in *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, (Split, Croatia), pp. 289–292, IEEE, Sept. 2014.

[75] C. Jennings, Z. Shelby, and J. Arkko, “IETF Draft - Media Types for Sensor Markup Language,” draft-jennings-core-senml-10, IETF, 2012.

[76] M. B. Alaya, S. Medjiah, T. Monteil, and K. Drira, “Toward Semantic Interoperability in oneM2M Architecture,” *IEEE Commun. Mag.*, vol. 53, no. 12, pp. 35–41, 2015.

[77] IPSO, “IPSO Alliance Framework.” http://www.ipso-alliance.org/wp-content/media/draft-ipso-app-framework-04.pdf.

[78] E. Kovacs *et al.*, “Standards-Based Worldwide Semantic Interoperability for IoT,” *IEEE Communications Magazine*, vol. 54, no. 12, pp. 40–46, 2016.

[79] A. Haller *et al.*, “The Modular SSN Ontology: A Joint W3C and OGC Standard Specifying the Semantics of Sensors, Observations, Sampling, and Actuation,” *Semantic Web*, vol. 10, no. 1, pp. 9–32, 2019.

[80] D. Pfisterer *et al.*, “SPITFIRE: Toward a Semantic Web of Things,” *IEEE Communications Magazine*, vol. 49, no. 11, pp. 40–48, 2011.

[81] O. Novo and M. D. Francesco, “Semantic Interoperability in the IoT: Extending the Web of Things Architecture,” *ACM Trans. Internet Things*, vol. 1, no. 1, pp. 6:1–6:25, 2020.

[82] M. Amoretti, O. Alphand, G. Ferrari, F. Rousseau, and A. Duda, “DINAS: A Lightweight and Efficient Distributed Naming Service for All-IP Wireless Sensor Networks,” *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 670–684, 2017.

[83] C. Hesselman *et al.*, “The DNS in IoT: Opportunities, Risks, and Challenges,” *IEEE Internet Computing*, vol. 24, no. 4, pp. 23–32, 2020.

[84] P. Brunisholz, F. Rousseau, and A. Duda, “DataTweet for user-centric and geo-centric IoT communications,” in *Proceedings of the 2nd Workshop on Experiences in the Design and Implementation of Smart Objects - SmartObjects ’16*, (New York City, New York), pp. 29–34, ACM Press, 2016.

[85] G. Niemeyer, “Geohash,” 2008.

[86] G. M. Morton, *A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing*. IBM, 1966.

[87] Google, “Plus Codes: Addresses for Everyone,” 2020.

[88] Google, “Open Location Code,” 2020.

[89] A. Rahman and E. Dijk, “Group Communication for the Constrained Application Protocol (CoAP),” Request for Comments RFC 7390, Internet Engineering Task Force, Oct. 2014.

[90] IEEE Standards Association, “Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID),” 2018.

[91] S. Cheshire and M. Krochmal, “DNS-Based Service Discovery,” 2013.

[92] M. Skwarek, M. Korczyński, W. Mazurczyk, and A. Duda, “Characterizing Vulnerability of DNS AXFR Transfers with Global-Scale Scanning,” in *Proc. of the IEEE Security and Privacy Workshops*, 2019.

[93] L. Song, "dns2 - A DNS Server and Client Implementation," 2020.

[94] R. Perdisci, I. Corona, D. Dagon, and W. Lee, "Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces," in *2009 Annual Computer Security Applications Conference*, pp. 311–320, Dec. 2009.

[95] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware," in *21st {USENIX} Security Symposium ({USENIX} Security 12)*, pp. 491–506, 2012.

[96] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive DNS analysis service to detect and report malicious domains," *ACM Transactions on Information and System Security*, vol. 16, pp. 1–28, Apr. 2014.

[97] W. Wang and K. Shirley, "Breaking Bad: Detecting malicious domains using word segmentation," *arXiv:1506.04111 [cs]*, June 2015.

[98] M. Korczynski, M. Wullink, S. Tajalizadehkhoob, G. C. M. Moura, A. Noroozian, D. Bagley, and C. Hesselman, "Cybercrime After the Sunrise: A Statistical Analysis of DNS Abuse in New gTLDs," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, (Incheon Republic of Korea), pp. 609–623, ACM, May 2018.

[99] V. Le Pochat, T. Van Goethem, and W. Joosen, "A Smörgåsbord of Typos: Exploring International Keyboard Layout Typosquatting," in *2019 IEEE Security and Privacy Workshops (SPW)*, (San Francisco, CA, USA), pp. 187–192, IEEE, May 2019.

[100] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a Dynamic Reputation System for DNS," p. 17.

[101] N. Kheir, F. Tran, P. Caron, and N. Deschamps, "Mentor: Positive DNS Reputation to Skim-Off Benign Domains in Botnet C&C Blacklists," in *ICT Systems Security and Privacy Protection* (N. Cuppens-Boulahia, F. Cuppens, S. Jajodia, A. Abou El Kalam, and T. Sans, eds.), vol. 428, pp. 1–14, Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[102] P. Lison and V. Mavroeidis, "Neural reputation models learned from passive DNS data," in *2017 IEEE International Conference on Big Data (Big Data)*, pp. 3662–3671, Dec. 2017.

[103] V. L. Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation," *Proceedings 2019 Network and Distributed System Security Symposium*, 2019.

[104] M. Antonakakis, R. Perdisci, W. Lee, N. V. Ii, and D. Dagon, "Detecting Malware Domains at the Upper DNS Hierarchy," p. 16.

[105] S. Maroofi, M. Korczynski, C. Hesselman, B. Ampeau, and A. Duda, "COMAR: Classification of Compromised versus Maliciously Registered Domains," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, (Genoa, Italy), pp. 607–623, IEEE, Sept. 2020.

[106] Crime Complaint Center (IC3), FBI, "Internet Crime Report." https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf, 2020.

[107] S. Kitterman, "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1," Request for Comments RFC 7208, Internet Engineering Task Force, Apr. 2014.

[108] M. Kucherawy, D. Crocker, and T. Hansen, "DomainKeys Identified Mail (DKIM) Signatures," Request for Comments RFC 6376, Internet Engineering Task Force, Sept. 2011.

[109] M. Kucherawy and E. Zwicky, "Domain-based Message Authentication, Reporting, and Conformance (DMARC)," Request for Comments RFC 7489, Internet Engineering Task Force, Mar. 2015.

[110] S. Hao, M. Thomas, V. Paxson, N. Feamster, C. Kreibich, C. Grier, and S. Hollenbeck, "Understanding the domain registration behavior of spammers," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, (Barcelona Spain), pp. 63–76, ACM, Oct. 2013.

[111] S. Maroofi, M. Korczynski, and A. Duda, "From Defensive Registration to Subdomain Protection: Evaluation of Email Anti-Spoofing Schemes for High-Profile Domains," p. 9.

[112] S. Maroofi, M. Korczyński, A. Hölzel, and A. Duda, "Adoption of email anti-spoofing schemes: A large scale analysis," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3184–3196, 2021.

[113] C. T. Deccio, T. Yadav, N. Bennett, A. Hilton, M. Howe, T. Norton, J. Rohde, E. Tan, and B. Taylor, "Measuring email sender validation in the wild," in *CoNEXT*, ACM, 2021.

[114] SpamHaus, "The SpamHaus Project." https://www.spamhaus.org.

[115] SURBL, "SURBL - URI reputation data." http://www.surbl.org.

[116] J. Bayer, Y. Nosyk, O. Hureau, S. Fernandez, I. Paulovics, A. Duda, and M. Korczyński, "Study on Domain Name System (DNS) Abuse Appendix 1 – Technical Report," tech. rep., 2022.

[117] F. Weimer, "Passive DNS Replication." https://www.first.org/conference/2005/papers/florian-weimer-paper-1.pdf.

[118] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. E. Hoffman, "Specification for DNS over Transport Layer Security (TLS)," 2016.

[119] P. E. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," 2018.

[120] Weka 3, "Machine Learning Software in Java."

[121] Pedregosa, F., *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[122] M. Zhu, J. Xia, X. Jin, M. Yan, G. Cai, J. Yan, and G. Ning, "Class weights random forest algorithm for processing class imbalanced medical data," *IEEE Access*, vol. 6, pp. 4641–4652, 2018.

[123] M. Korczyński, S. Tajalizadehkhoob, A. Noroozian, M. Wullink, C. Hesselman, and M. Van Eeten, “Reputation Metrics Design to Improve Intermediary Incentives for Security of TLDs,” in *IEEE EuroS&P*, 2017.

[124] ICANN, “Temporary Specification for gTLD Registration Data,” May 2018.

[125] S. Marchal, G. Armano, T. Grondahl, K. Saari, N. Singh, and N. Asokan, “Off-the-Hook: An Efficient and Usable Client-Side Phishing Prevention Application,” *IEEE Transactions on Computers*, vol. 66, pp. 1717–1733, Oct. 2017.

List of Figures

List of Tables