Electronics and IT
Aalborg University

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Domain Security: A Study of Newly Registered Domains and their relation to Malicious Domains.

Theme:
Master Thesis

Project Period:
Autumn - Spring 2023-2024

Participant(s):
Chrysoula Katsika
Severen Joyton Fernandes

Supervisor(s):
Marios Anagnostopoulos

Copies: 1

Page Numbers: 55

Date of Completion:
30 May 2024

Abstract:

The Domain Name System (DNS) is a fundamental component of internet infrastructure, essential for translating human-readable domain names into IP addresses. This thesis delves into the security threats posed by Newly Registered Domains (NRD's) and their association with malicious activities within the cyber realm. By analyzing a comprehensive dataset of NRD's, the thesis aims to identify discernible patterns and behaviors that could indicate malicious intent, thereby identifying potential threats. Employing DNS-based features, entropy analysis, and graphical representations using Neo4j the research aims to uncover and characterize key traits exhibited by malicious domains. The methodology leverages the use of DNS-based features and entropy analysis to differentiate between benign and malicious domains. Furthermore, graphical representations using Neo4j elucidate the relationships and patterns found within the dataset. The results reveal a range of behaviors that can signal malicious intent, providing valuable insights into the characteristics and behaviours of harmful domains. The result contributes significantly to a deeper understanding of malicious domain behaviors, providing a foundation for the development of more effective detection and prevention strategies. Strengthening proactive monitoring mechanisms and implementing rigorous verification processes emerge as imperative measures for mitigating the associated risks posed by such domains.

Acknowledgements

We would like to begin by expressing our most sincerest gratitude to Aalborg University of Copenhagen and Professor Jens Myrup Pedersen for letting us be a part of the study in Cybersecurity and the community. To our supervisor Dr. Marios Anagnostopoulos whose guidance, expertise, and unwavering support have been invaluable throughout this research journey, a big thank you. We are immensely grateful for his encouragement and knowledge given to us during the two years of the Masters Study.

Chrysoula
I would like to thank my co-author, Severen Fernandes, for his partnership and dedication throughout this project. Your insights, hard work, and enthusiasm have been invaluable. I am deeply grateful for your patience and collaboration, which made this process both rewarding and enjoyable. Moreover, I want to express my deepest appreciation to my parents and my sister for their endless love and support throughout my two years in this master's program. Without you, I would not be the person I am today. Your encouragement, love, and belief in my abilities have been my anchor during challenging times. Furthermore, I would like to express my gratitude to my very good friends Ilianna, Stavroula, Theofilos, and Vasileios for believing in me during times when I did not believe in myself. Your belief has been a constant source of motivation and strength and I am deeply grateful for your friendship and trust. Thank you for being my pillars of support and for always cheering me on.

Severen
I would like to begin by extending my heartfelt gratitude to my thesis partner Chrysoula Katsika, thank you for your patience and understanding, always available to brainstorm ideas, offer helpful critiques, and provide moral support during challenging times. I am profoundly grateful to my family for their constant love, support, and encouragement throughout this journey. Your patience, understanding, and for always being there when I needed you. Your sacrifices and support have made this accomplishment a reality, and I am forever indebted to you. I would like to dedicate this paper in honor of my late father, whose unwavering belief in me has been the foundation of my success. I would like to express my sincere appreciation to my friends from India, London and Denmark, for their unwavering support and encouragement throughout this journey. Your words of motivation, understanding, and companionship have been invaluable. Thank you for always being there for me, regardless of the highs and the lows. I am thankful for your friendship, which has been a source of strength and inspiration, and I am truly grateful for your presence in my life.

Together we also want to acknowledge our colleagues and peers for their encouragement, support, and stimulating discussions, which have contributed not only to the intellectual vibrancy but also the sense of community that have made the journey of this academic endeavor enjoyable and memorable.

Copenhagen,DK
30/05/2024

Contents

Listings

Chapter 1

Introduction

The Achilles' Heel of the Internet, the Domain Name System (DNS), operates under a perpetual threat landscape, with no foreseeable end in sight as these dangers evolve. DNS primarily utilizes the User Datagram Protocol (UDP) for its functions, occasionally incorporating Transmission Control Protocol (TCP) as well. The inherent nature of UDP, lacking connections, renders the DNS protocol vulnerable, making it a favored target for Distributed Denial of Service (DDoS) attacks [1].

Recognized as the Internet's phonebook, DNS serves as a vital component of the global internet infrastructure, facilitating the translation between familiar names and the numerical addresses necessary for accessing websites and sending emails. However, its significance also makes it a prime target for malicious actors seeking to compromise corporate and sensitive data, as evidenced by the escalating threat levels highlighted in recent warnings.

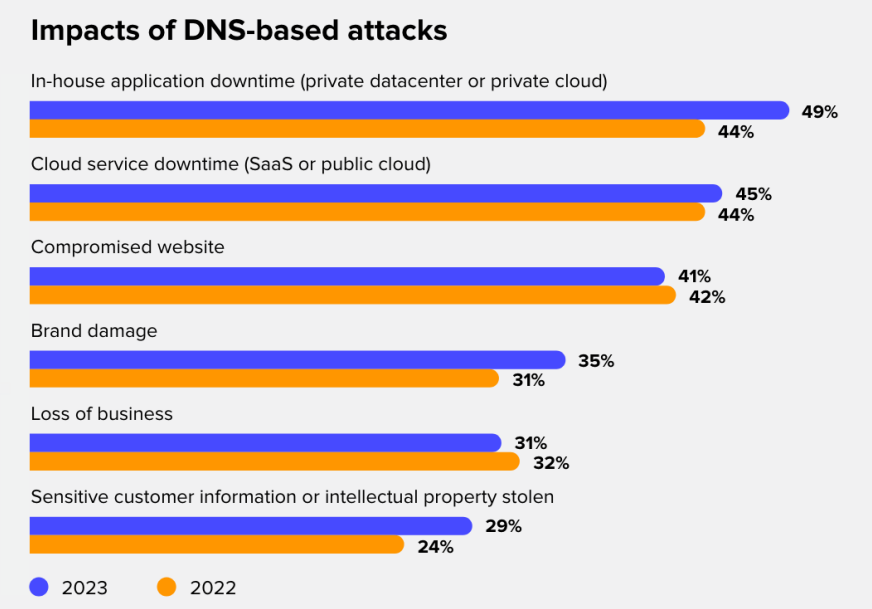


Figure 1.1: Surge in DNS attacks [2]

The "2023 IDC Global DNS Threat Report"[2], presents a bleak outlook, revealing a significant surge in DNS attacks compared to the preceding year 1.1. This critical internet infrastructure, responsible for translating website names into addresses, has become a focal point for cybercriminals. The financial repercussions are substantial, with the average cost of a DNS attack soaring by a staggering 49% as shown in figure 1.2. In the United States, the average cost exceeds $1,27 million per attack. Additionally, the report underscores concerning statistics regarding the severity of these attacks. Nearly half of the surveyed organizations suffered losses exceeding $500,000 from a single attack, with a troubling 10% experiencing losses surpassing $5 million.

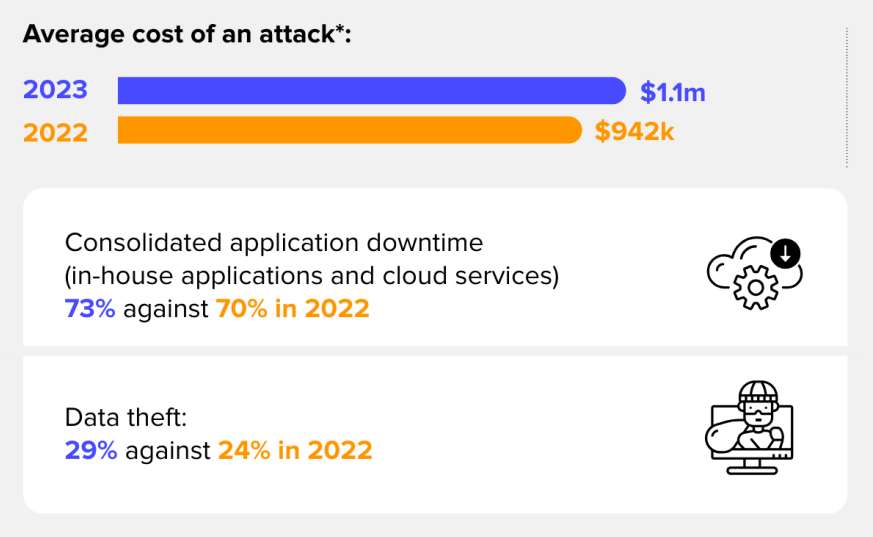


Figure 1.2: Cost of an attack [2]

Moreover, a considerable portion of US companies conceded that it took more than a day to detect a DNS attack. This slow detection rate indicates a worrying gap in cybersecurity measures. This vulnerability is partly attributed to the reliance on the UDP, which lacks robust security features and renders DNS susceptible to DDoS attacks. In comparison to the previous year's findings, these results suggest a worsening trend in DNS threats, emphasizing the critical need for organizations to fortify their DNS security measures[2].

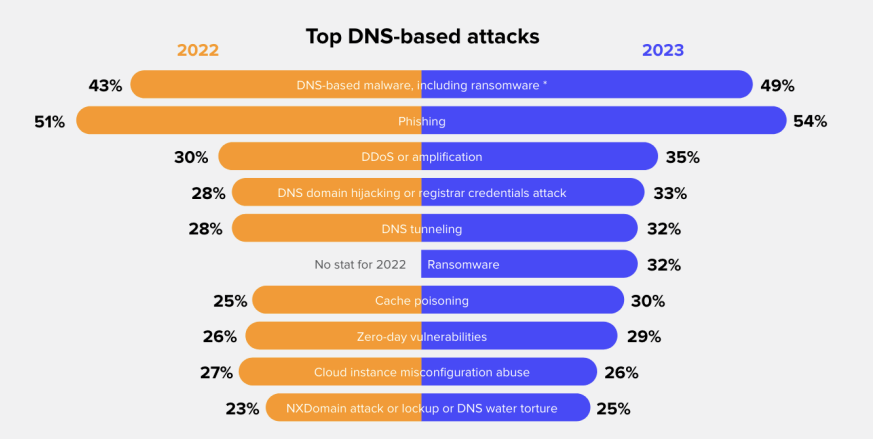


Figure 1.3: DNS in the Cybersecurity Landscape [2]

Examining the landscape of cyber threats as shown in figure 1.3 and the evolving tactics of malicious actors reveals a concerning trend: while DDoS attacks remain prevalent, the emergence of phishing and spam campaigns is on the rapid rise. These nefarious activities often find their origins in the exploitation of DNS infrastructure to generate and propagate malicious domain names. Such domains serve as launchpads for a plethora of cybercrimes, including hosting phishing websites, distributing malware, and orchestrating various forms of online fraud. Recent research underscores the pervasive nature of this threat, indicating that the average user encounters approximately 5,000 DNS queries daily, with an alarming rate of approximately one in 1,000 queries being identified as malicious. This translates to an astonishing average of 1,825 malicious queries per user annually.

Consequently, malicious domains represent a growing risk across all sectors of the digital landscape, with the DNS layer emerging as a primary vector for cybercriminal activities.

Fortunately, amidst the escalating tide of cyber threats, security professionals have access to a variety of tools and strategies to fortify their defenses. While newer top-level domains and specific hosting providers have been identified as common sources of malicious content, recent studies indicate that newly registered domains (NRD's) and free SSL certificates do not inherently pose higher risks. However, vigilance and proactive measures remain paramount in mitigating these threats effectively. DNS Filter's observations further underscore the severity of the issue, revealing a staggering 1,250% year-over-year surge in malicious domains registered within a 24-hour period.

In response to this surge in malicious activities, cybersecurity experts emphasize the necessity of a comprehensive approach that encompasses proactive threat detection and mitigation strategies. Dave Mitchell, CTO at Hyas, [3] highlights DNS as a favored mechanism for malware communication, underscoring the critical importance of safeguarding against such threats. The identification and assessment of domain reputation are crucial steps in breach prevention, demanding a multi-faceted approach aided by various tools and assessments. One prevalent countermeasure against malicious domains is the utilization of blocklists curated by reputation providers. These lists facilitate the swift identification and blocking of domains associated with malicious activities. However, while effective, this reactive approach often relies on post-attack reporting, necessitating a more proactive stance to stay ahead of emerging threats.

Proposals to address this challenge extend beyond reactive measures to encompass proactive interventions at the domain registration stage. By scrutinizing factors such as domain registration information and potential criminal affiliations, preemptive measures can be implemented to thwart malicious intent before it manifests. Such interventions could be executed at the Registrar or Registry level, where early detection and intervention can prevent abuse before it occurs, thus enhancing overall cybersecurity resilience.

1.1 Problem Statement

Given the persistent threat of DNS attacks, there is an urgent requirement for a comprehensive examination of newly registered domains and their potential ties to malicious entities. Despite numerous existing studies attempting such analysis, there remains a notable gap in understanding the intricate behaviors exhibited by malicious domains. Addressing this gap, this thesis aims to illuminate the following research questions:

- What are the patterns and behaviors of malicious domains?
- What are the common characteristics of malicious domains?
- How do these patterns and characteristics provide the researchers with evidences for identifying a potentially malicious domain even prior to its creation through a Newly-Registered Domain?
- How does Neo4j enhance the understanding of these patterns?

This paper endeavors to answer these questions through a comprehensive analysis of their behaviors and characteristics. Through the systematic collection and analysis of newly registered domains for over a period of four months from 1st of December 2023 to 31st of March 2024, this study aims to elucidate patterns indicative of malicious intent, thereby enhancing proactive threat mitigation strategies and bolstering cybersecurity defenses in an increasingly complex digital landscape.

1.2 Thesis structure

The structure of this thesis is based on how the project was created in more or less chronological order. The fundamental idea and rationale behind the project are to analyze the relationship between NRD's and malicious domains. In chapter 1 is the problem statement which the paper's cornerstone and primary emphasis. Followed by chapter 2 where the background of how DNS works. Chapter 3 describes some of the existing analysis that already had been done on registered domains. The experimental method based on specific features and the analysis patterns are presented in Chapter 4. Chapter 5 defines the results grounded in experimental method on a specific time period observing data. The discussion and evaluation based on the results are explained in Chapter 6. At the end of the paper, Chapter 7 provides a comprehensive overview of the project's journey and analyzes the limitations encountered during the research, and proposes directions for future work.

Chapter 2

Background

2.1 Domain Name System

DNS is the fundamental ingredient in the recipe of Internet Protocol (IP) communications, whose main functionality is to translate human readable domain names and map them to IP addresses. Even though this has remained as its primary and basic function, DNS today has evolved into a hierarchical and decentralized structure.

In order to understand the DNS as a whole we first need to understand the functionalities of two factors that form as the basis of the DNS, namely 1)DNS Servers and 2)DNS Protocols and Resource Records(RR),

2.1.1 DNS Servers

DNS is made of three major components, namely: (a)DNS Recursive Resolvers, (b)Root Nameserver servers, and (c)Authoritative Nameservers.

a) **DNS Recursive Resolvers:** According to RFC 1034 [4], recursive resolvers can be understood as a cohesive suite of software programs responsible for extracting information from name servers in response to client requests. To fulfill this task, resolvers are equipped with the ability to establish connections with at least one name server and leverage the data within that name server to directly address queries. In cases where the resolver encounters a query that necessitates additional information, it can intelligently navigate the resolution process by making referrals to other name servers. A resolver typically operates as a system routine that is readily accessible to user programs. This accessibility means that there is no need for a dedicated communication protocol between the resolver and the user program, simplifying the interaction between the two components and streamlining the data retrieval process. In essence, resolvers act as the intermediaries that facilitate the seamless exchange of data between the user's request and the DNS infrastructure, ensuring efficient and accurate information retrieval.

b) **Root Nameserver:** is the set of all domain names that are registered in the DNS. These domain names are organized into a tree-like structure, with the top of the tree being the root domain. Below the root domain, there are a number of top-level domains (TLDs), such as '.com', '.net', and '.org'. This nameserver is the first step in the search for a specific DNS query, and it hosts the right-most lable of a domain name. For example, in the domain name 'google.com', '.com' is the TLD. Some other popular TLD's include '.org', '.uk', and '.edu' [5].

   TLD's play an important role in the DNS lookup process. For all uncached requests, when a user enters a domain name like 'google.com' into their browser window, the DNS resolvers start the search by communicating with the root nameserver which points to the TLD server. In this case, the TLD is '.com', so the resolver will contact the TLD DNS server, which will

then provide the resolver with the IP address of Google's origin server [6]. Each domain name in the DNS name space corresponds to a set of Resource Records (RRs), which contain information about that domain name, such as its IP address, mail servers, and other information discussed in the following subsection. The DNS name space is hierarchical, meaning that each domain name can have subdomains beneath it. For example, the domain name "example.com" could have subdomains such as "www.example.com" and "mail.example.com". The domain name space is managed by a number of organizations, including the Internet Corporation for Assigned Names and Numbers (ICANN). DNS who is responsible for the top-level domains. The domain name space is an essential part of the Internet. It allows us to easily access websites and other internet resources without having to memorize IP addresses.

c) **Authoritative Nameservers:** A type of DNS server that stores and maintains all DNS records for a domain, including 'A' records, 'MX' records, or 'CNAME' records. Almost all domains rely on multiple nameservers to increase reliability for example, if one nameserver goes down or is unavailable, DNS queries can go to another one. Typically, there is one primary nameserver and several secondary nameservers, which store exact copies of the DNS records in the primary server. Updating the primary nameserver will bring an update to the secondary nameservers as well. When multiple nameservers are used during a query (as in most cases), 'NS' records should list more than one server [7].

2.1.2 DNS Protocols and Resource Records:

While DNS servers handle the task of translating domain names into IP addresses, with recursive servers executing queries for clients and authoritative servers furnishing definitive responses for particular domains, the DNS protocol and resource records collectively facilitate this operation within a decentralized framework, best described as:

- **DNS Protocol:** DNS protocol is a decentralized system used to translate human-readable domain names into machine-readable IP addresses. It facilitates the mapping of domain names to IP addresses and vice versa. The DNS protocol operates through a distributed network of DNS servers, which work together to resolve DNS queries and maintain the DNS namespace. The DNS protocol includes various components such as DNS queries, DNS responses, DNS records, and DNS servers, which collaborate to ensure efficient and reliable domain name resolution.

- **Resource Records:** RR's are the fundamental building blocks of the DNS database. They contain information about various types of data associated with domain names, such as IP addresses, aliases, mail server addresses, and more. There are many types of resource records, each serving a specific purpose. Some common types of RR's are depicted in figure 2.1. Resource records are used in DNS responses to provide information about domain names, such as IP addresses, mail server addresses, and other types of data which are further discussed in Chapter 4 of the paper.

This sets the stage for delving into the intricacies of the DNS architecture, offering insight into the components that underpin the system's functionality. By comprehending these components, we gain a clearer understanding of the intricate process behind a DNS lookup. This process will be explored further in the subsequent segment of this paper, elucidating the steps involved in resolving domain names to their corresponding IP addresses.

TABLE III
STRUCTURE AND PRINCIPAL TYPES OF A RESOURCE RECORD (RR)

RR field	Description
NAME	Encodes the name of the node which this record pertains to.
TYPE	Gives info about the kind of data stored in the RDATA field of this record.
CLASS	Indicates the class this record belongs to. Usually, RRs belong to the Internet (IN) class.
TTL	The time interval that this RR may be cached before the source of the information should again be consulted.
RDLENGTH	Specifies the byte length of the data stored in the RDATA field.
RDATA	A variable length string of octects encoding the actual RR data, whose internal format varies with the TYPE and CLASS fields.
RR type	**Description**
A	A 32-bit IPv4 address typically encoding an host address, but which is also used for other purposes (e.g., storing subnet masks).
AAA	A 128-bit IPv4 address used to encode an host address.
CNAME	An alias for a domain name that specifies the primary (or canonical) name for the owner.
DNAME	Alias for a name and all its subnames (CNAME is an alias for only the exact name).
MX	Specifies a domain name of a host willing to act as a mail exchange for the owner name.
NS	A name server that is supposed to be authoritative for the given class and domain.
PTR	Used to point to another location in the domain name space.
SOA	Used to describe authoritative information about the given zone.
TXT	Record originally introduced for human-readable text but more often used to store machine-readable data.

Figure 2.1: Types of Resource Records (RR) [8]

2.1.3 DNS Lookup

Based on the above two functionalities, the process of DNS resolution is achieved through relating a host-name (such as www.example.com) into a computer-friendly IP address (such as 192.XXX.X.X). Each device linked to the Internet is given an unique IP address, which helps in locating the required device. When a user loads to a webpage, the input given to a web browser which is user-friendly ("example.com") is translated to a machine-friendly address necessary to locate the "example.com" webpage.

DNS is responsible for translating human-friendly domain names like "example.com" into the numerical IP addresses that computers use to communicate. To understand this process, let's examine the DNS lookup process, which consists of eight steps, as depicted in Figure 2.2. When a user enters a domain name into a web browser or other application, their request first reaches a recursive resolver, which plays a crucial role in navigating the DNS hierarchy. In iterative mode, the resolver interacts with different DNS servers along the query path, each providing a piece of the puzzle, until the ultimate IP address is discovered. Alternatively, recursive mode allows the resolver to handle the entire query on behalf of the user, traversing the DNS architecture to retrieve the

desired IP information.

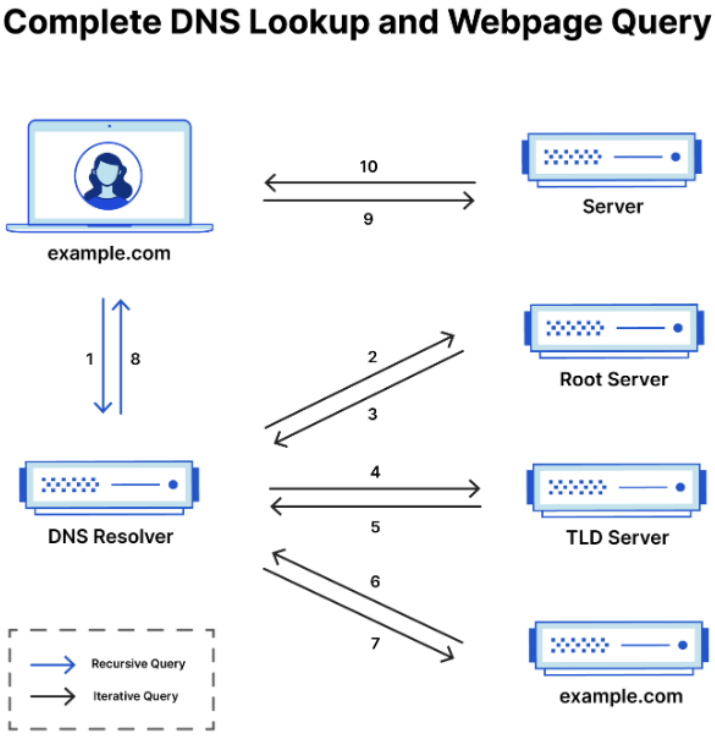


Figure 2.2: DNS Lookup [9]

If the user needs to gather information about the entire DNS path, including various references, an anonymous recursive method is employed. Forwarding resolvers, on the other hand, provide a more streamlined approach. They handle DNS queries by either retrieving the information from a local cache or forwarding the request to a recursive resolver. Designed for small-scale networks, forwarding resolvers can even utilize the "/etc/hosts" file to resolve local hostnames that are not included in the global DNS database. Hosts files, which are plain text files, serve as a local mapping of fully qualified domain names (FQDNs) to the corresponding server IP addresses. These files are particularly useful when a DNS server is unavailable and a user attempts to access a domain through their browser. In such cases, Linux relies on the "/etc/hosts" file to resolve the domain name.

The hierarchy of the DNS is based on a distributed database organizational structure, dividing the database into manageable zones. Every zone is allocated to an authority with complete power to oversee the records within one or more domains [8]. This results in the DNS to form an inverted tree hierarchy, where the authority or root is at the top. IANA (Internet Assigned Numbers Authority), a department of ICANN is responsible for managing the root of the DNS hierarchy. It oversees the allocation of generic TLD's, such as '.com', '.org', and country-code TLD's like '.us' or '.uk'. IANA

also manages the root zone file, which contains authoritative information about the root of the DNS tree. Going down a level, we have the Registry who is the authoritative database that maintains and manages the registration information for a specific top-level domain (TLD). It is responsible for the overall administration of domain names within a particular TLD's or country-code TLD's. The registry holds the master database of all registered domain names under its TLD and manages the associated DNS records for those domains. Examples of domain registries include "Verisign" for '.com', "Public Interest Registry" for '.org', and "Nominet" for '.uk'.

Making an entry into the registry is usually done with the help of Registrars, an accredited organization or company that acts as an intermediary between individuals or businesses (registrants) and the domain name registry. Registrars are authorized by domain registries to offer domain registration services to the public. They facilitate the process of domain registration, renewal, and management on behalf of domain owners. Registrars interact with the registry to update and maintain the registration information in the registry's database. Examples of domain registrars include "GoDaddy", "Namecheap", and "Google Domains".

This leads to the understanding of how domains are stored and how the resolver starts at the root to gain the TLD of the server that stores the information of its domains. When searching for example.com, the request is pointed towards the '.com' DNS, to which the DNS responds with the IP address of the domain's nameserver, like "example.com".

When you type a domain name into your browser, a request is sent to your internet service provider's (ISP) DNS servers. These servers typically have cached copies of IP address information, so they can often provide you with the IP address you need right away. However, in its quest for the current IP address if the cached information is outdated or does not exist, the ISP's servers will need to query the domain nameserver that is also known as the authoritative DNS server. An authoritative DNS server is the final source of truth for a domain name's IP address. These servers are maintained by the domain's owner and contain the most up-to-date information. When an ISP's DNS server asks an authoritative DNS server for an IP address, the authoritative server will either provide the answer directly or redirect the request to another authoritative server that is responsible for the domain. Authoritative DNS servers are designed to be efficient and fast, and they will not respond to recursive queries. Recursive queries are requests from other DNS servers that are trying to find the IP address of a domain. Instead, authoritative servers only respond to requests from resolvers, which are the first DNS servers to receive a query from a user. This system ensures that users are always able to access the correct IP address for a domain, even if the cached information is out of date. It also allows for efficient routing of DNS requests, as resolvers only need to contact authoritative servers when they can't find the information they need elsewhere.

Following the response returned by the authoritative nameserver of the domain, the DNS resolver then responds to the web browser with the IP address of the domain requested initially. To which leads to the end of the quest in querying the location of the requested domain name and the web browser is able to use the returned IP address to make a request to the webpage like "example.com". This culminates and gives a overview on how the DNS works.

2.2 Domain Name System Security Extension

DNS, formulated in the 1980s during a time when the Internet was considerably smaller, did not prioritize security in its initial design. Consequently, when a recursive resolver sends a query to an authoritative name server, there existed no means to authenticate the legitimacy of the response. The resolver's sole method of verification was that the response seemed to originate from the same

IP address to which the resolver sent the initial query. However, relying on the source IP address as an authentication mechanism was weak since the source IP address of a DNS response packet can be effortlessly built or spoofed. In its original design, DNS lacked the capability for a resolver to easily discern a counterfeit response to one of its queries. A malicious actor could readily impersonate the authoritative server originally queried by faking a response that gave the appearance of originating from that authoritative server. In essence, this permitted the malicious actor to redirect a user to a potentially malicious website without the user's awareness [10].

Internet Engineering Task Force (IETF), which is responsible for the DNS protocol standards, realized that the lack of stronger authentication in DNS was a major problem. The solution that resulted was the Domain Name System Security Extensions (DNSSEC). According to ICANN [10], DNSSEC strengthens authentication in DNS using digital signatures based on public key cryptography. With DNSSEC, it is not the DNS queries and responses themselves that are cryptographically signed, but rather the DNS records that are signed by the owner of the data [11].

DNSSEC enhances the security of the domain name system by introducing cryptographic signatures to the pre-existing DNS records. These digital signatures are integrated into DNS name servers, coexisting with standard record types such as 'A', 'AAAA', 'MX', 'CNAME', and more [12]. DNSSEC introduces two crucial functionalities to the DNS protocol [10]:

- Data origin authentication empowers a resolver to cryptographically confirm that the data it receives indeed originates from the expected zone, thereby ensuring the authenticity of the information.

- Data integrity protection provides the resolver with the assurance that the data has remained unaltered during transit, safeguarding its integrity from the moment it was initially signed by the zone owner using the zone's private key.

DNSSEC plays a crucial role in safeguarding against a diverse range of threats, including DNS spoofing, Man-in-the-Middle attacks, and cache poisoning. As essential services increasingly rely on the DNS, ensuring security becomes paramount. Despite the growing significance of DNSSEC in the evolving landscape of the Internet, it is not obligatory when registering a new domain name. This lack of mandatory enforcement contributes to the occurrence of registering new domains for malicious purposes such as to re-register or re-purpose. NRD's with disabled DNSSEC are especially susceptible to various attacks, such as DNS hijacking, cache poisoning, and domain squatting. Among these threats, DDoS attacks on NRD's are common.

In 2015, nearly half a million Alabama cell phone users received identical text messages prompting them to click a link for purported bank account verification. The provided link led to a convincing fake bank website, where unsuspecting users entered their sensitive financial details [13]. Two years later, a Dutch security firm fell victim to a sophisticated attack, enabling hackers to seize control of its servers and intercept clients login credentials and confidential data. Unauthorized access to the firm's account with a third-party domain registrar allowed attackers to manipulate a domain name system record, effectively hijacking control of the firm and all incoming traffic [14].

2.3 Malicious Domain Names

Domain names play a crucial role as the internet's phone book, serving as vital tools for users seeking access to online goods and services. Businesses and organizations consider domain names as valuable assets for their brands. Every day, thousands of NRD's emerge, with many serving legitimate purposes like introducing new products, establishing new websites, or building new brands.

Nevertheless, the prevailing majority of these NRD's raise suspicion, and a significant portion of them harbor malicious intent [15].

In the ever-evolving realm of cybercrime NRD's, characterized by their recent creation or ownership change within 32 days as per the regulations of the ICANN [16], often masquerade as legitimate brand websites, leveraging subtle variations in domain names to deceive unsuspecting users. Their short lifespans, often spanning mere days or weeks, pose significant challenges for detection and mitigation. The ease of domain registration, requiring minimal expertise, further amplifies the risk posed by NRD's. Corporate security systems often overlook these recently created domains, granting cybercriminals free rein to execute their malicious campaigns. Among their nefarious activities, DDoS attacks against organisations through NRD's stand out. These attacks aim to flood targeted websites or servers with excessive traffic, rendering them inaccessible to legitimate users. Newly registered domains also serve as conduits for malware distribution, including viruses, worms, and trojans. Attackers embed malware within website content or disguise malicious links within seemingly innocuous emails. Clicking on these links or downloading infected files unknowingly installs malware on users' devices, enabling attackers to establish command-and-control channels for remote manipulation, data theft, and launching further cyber assaults [17].

The criticality of swift detection and proactive security measures cannot be overstated. Malicious NRD's have a long history of wreaking havoc in the cyber world. In 2019, a group known as Magecart infiltrated the websites of major retailers and e-commerce platforms, including "British Airways", "Ticketmaster", and "Sephora". By injecting malicious JavaScript code into these websites, they intercepted customer data as it was being entered on checkout pages. This combination of compromised websites and malicious NRD's resulted in a massive trove of credit card information being compromised, affecting millions of customers worldwide.

Another infamous example of NRD exploitation involved the 'ZeuS' botnet, a large-scale network of infected computers that spread malware, stole banking credentials, and launched DDoS attacks. The botnet's operators controlled their malicious empire through a network of NRD, communicating with infected machines and receiving stolen data. The proliferation of newly registered domains necessitates a concerted effort to enhance cybersecurity measures. Employing advanced detection techniques, employing reputable domain registrars with stringent security protocols, and educating users to be wary of suspicious emails and websites are crucial steps towards mitigating the risks posed by NRD's. Newly registered domains have emerged as a formidable tool in the arsenal of cybercriminals, posing significant threats to online security. Their ease of creation, deceptive nature, and short lifespans make them difficult to detect and mitigate. Swift detection, proactive security measures, and user education are essential for combating the growing menace of NRD's and safeguarding the integrity of the digital landscape [15].

2.4 Domain Name Vulnerability

A way a domain can be made malicious is through Domain name vulnerabilities which refers to weaknesses or susceptibilities associated with the management, registration, or operation of domain names, which can be exploited by malicious actors to compromise the security, availability, or integrity of a domain or its associated services. These vulnerabilities can manifest in various forms, including:

- **Domain Hijacking:** Domain hijacking occurs when unauthorized individuals gain control over a domain by exploiting weaknesses in domain registrar security or by compromising the credentials of the domain owner. This is essentially internet identity theft, as the original

owner loses control over their website content, email, and any other services relying on the domain name. Such incidents pose a serious threat to an organization's brand and reputation. Experts at major cybersecurity firms, including "Tripwire", "FireEye", and "Mandiant", have reported an alarming surge in DNS hijacking attacks worldwide since 2017. These attacks have targeted government, telecom, and internet entities across the Middle East, Europe, North Africa, and North America [18].

- **DNS Spoofing or Cache Poisoning:** DNS spoofing, also known as cache poisoning, targets DNS resolvers that cache commonly or recently requested DNS records. In this type of attack, malicious actors manipulate the DNS cache or DNS responses to redirect users to malicious websites or intercept their communications. One notable example is the "Forgot Password" cache poisoning attack. Vulnerabilities discovered in July 2021 revealed that 'forgot password' links in web applications were susceptible to such attacks. Security researchers found that by executing a cache poisoning attack on 146 vulnerable web applications, they could redirect password reset emails to attacker-controlled servers. This allowed attackers to click on the reset link and change the user's password, granting them legitimate access to the account [18].

- **Typo-squatting:** Typo-squatting is a form of cyber attack where malicious actors register domain names similar to popular or legitimate domains, often differing by just one or two characters. The goal is to exploit common typing errors made by users, redirecting them to malicious websites. One well-known real-life example involved the domain "www.paypai.com" (with an "i" instead of an "l"). Users intending to visit PayPal's website ("www.paypal.com" but mistakenly typing "paypai" were redirected to a fraudulent site mimicking PayPal's login page. The continued prevalence of typo-squatting was recently demonstrated by a worrying spike in Bifrost Linux malware variants over the past few months, using fake VMware domains [19].

- **DNSSEC Misconfiguration:** Incorrectly configuring DNSSEC can lead to vulnerabilities such as improper key management or failure to validate DNS responses, potentially exposing users to DNS-based attacks. Despite the security enhancements provided by DNSSEC, attackers can still exploit certain weaknesses. Understanding these vulnerabilities and implementing appropriate mitigation strategies is essential for maintaining the integrity and security of DNSSEC deployments. In 2012, several major Swedish banks faced targeted DNS-based attacks, redirecting users to malicious websites. These attacks prompted increased adoption of DNSSEC in Sweden's financial sector. Banks and financial institutions recognized the importance of DNSSEC in ensuring the authenticity and integrity of their online services, leading to widespread implementation of DNSSEC to protect against similar attacks in the future [20].

To mitigate domain name vulnerabilities, organizations should adopt best practices such as using strong authentication mechanisms for domain management accounts, regularly monitoring domain registrations and DNS configurations, implementing DNSSEC where applicable, educating users about phishing and social engineering threats, and promptly addressing any detected vulnerabilities or suspicious activities related to domain names. Next, we will dive into the related work section of the paper to explore the contributions of various studies that support our thesis.

Chapter 3

Related Work

The domain name, an integral component of the Domain Name System (DNS), remains susceptible to exploitation despite being its foundational element. Numerous studies have proposed methods to identify, restrict, and mitigate DNS abuse. However, a notable gap exists in security protocols, particularly concerning the safeguarding of the registration and usage of newly created domains. This literature review aims to delineate and highlight various aspects of the DNS architecture that facilitate the identification of malicious domains. While conventional approaches for detecting malicious domains primarily focus on the specific malevolent activities they engage in, this paper takes a unique approach. It seeks to analyze behavioral patterns using extracted features to assess whether a domain exhibits malicious or benign characteristics at the moment of its creation.

The literature surveyed in this section, spanning from 2014 to the present, offers insights into features commonly referred to as Resource Records (RR) associated with a domain name. These RR features play a pivotal role in the analysis and findings presented in this report.

Looking into the comprehensive overview of the current landscape of DNS security and privacy Schmid [8], discusses the historical evolution of the DNS infrastructure over the past thirty years, highlighting the trans-formative changes it has undergone. Emphasis is placed on the profound impact of security breaches and abuses related to DNS on businesses and citizens, and the different types of DNS-related threats. Additionally, the study categorizes and describes the different attacks that can affect the proper functioning of DNS, as well as attacks that rely on DNS to be exploited. Furthermore, the paper discusses the most relevant protocols introduced so far aimed at safeguarding communication among the name servers and the solutions introduced to protect the resolver subsystem. In a forward-looking approach, the author presents more radical alternatives that depart from both the conventional DNS resolution process and its reverse-tree shaped hierarchy of authorities. The paper concludes with a comparative analysis of the proposed solutions and an attempt to give some insight on the future of the Internet name service.

In a different perspective, Lyu et al. [21], contribute valuable insights into how DNSSEC was proposed for protecting the data integrity of DNS by providing cryptographic verification using digital signature so as to validate records given in a DNS response from the authoritative DNS server. The survey explores standard techniques like DoT, DoH, and DoQ, evaluating their current status and performance across the Internet. It also addresses the potential misuse of DNS encryption by malware for command and control and data exfiltration [22]. Furthermore, the study delves into methods for detecting encrypted DNS traffic, emphasizing the importance of identifying malicious encrypted DNS communications and profiling host behaviors. In conclusion they highlighted the security benefits and risks of DNS encryption techniques. Among the risks they focused mainly on misuse of encrypted DNS protocols by malware through the way of C&C and data exfiltration.

The survey by Zhauniarovich et. al. [23], offers a meticulous analysis of the role of DNS in detecting malicious domains and preventing attacks over the Internet. It provides a comprehensive overview of the various components required to implement a DNS-based detection technique, including DNS data collection, feature extraction, and machine learning algorithms. The paper also categorizes existing approaches based on various viewpoints, such as the type of features used, the type of machine learning algorithms employed, and the type of data sources used. The authors have

compiled a comprehensive bibliography of relevant papers and have carefully studied each paper to extract information that could help cover the targeted research topic. The paper also addresses the challenges faced by the research community in fully utilizing DNS data analysis to fight against the attacks, such as the need for accurate and up-to-date enrichment data. The survey concludes by highlighting the importance of DNS data analysis in detecting and preventing attacks over the Internet and by outlining future research directions in this area.

The paper by Hao et. al. [24], explores the potential of monitoring the DNS behavior of newly registered domains to detect potential malicious activity. The authors examine three specific features of DNS behavior that may be indicative of malicious activity: the number of IP addresses associated with a domain, the number of DNS queries made for a domain, and the time between domain registration and the first DNS query. Using a large dataset of '.com' and '.net' domains provided by Verisign, the authors find that these features can be used to distinguish between malicious and legitimate domains with a high degree of accuracy. The authors suggest that further research is needed to develop more sophisticated detection systems based on these findings. The study also addresses some limitations of previous work in this area, such as the use of partial datasets and the lack of clarity around the representation of the data.

The study by Al Messabi et. al. [25], describes a system for detecting malware using DNS records and domain name features. The system employs a Python script to extract and classify malicious domain names' features and then utilizing nslookup to find the IP addresses of the malicious domains. The system is designed to detect malicious domain names by observing the obvious features of suspicious domains and combining the features with some of the DNS-based attributes. They also describe the eight unique features used to detect malicious domain names, which include the length of the domain name, the number of digits in the domain name, the number of hyphens in the domain name, and the number of subdomains in the domain name. The system was implemented using real-world data in an experiment, and the results showed that it accurately identified malicious websites before they were visited. The potential benefits of using DNS for malware detection include reducing the economic impact of cybercrime on the global Internet economy.

The paper by Yang et. al. [26], examines the behavior of the Domain Name System (DNS) and its impact on the wider Internet. The paper explores DNS behavior from various perspectives, including query types, recursive resolvers, TTL's, hosting infrastructures, and query failures. With finding that 13.5% of DNS queries fail, and exploring into the root causes of these failures, the study identifies significant differences between IPv4 and IPv6 lookups, biased failure distribution across domains, the great impact of recursive resolvers and malicious domains on query failures. The study also highlights some interesting findings, such as the growing prevalence of public resolvers, which account for 13.5% of the total DNS requests in the dataset. However, the authors note that public resolvers differ in localization performance, and they recommend fine-tuning the TTL's for 'A' and 'AAAA'records to improve DNS performance.

Spooren et. al. [27], present a solution for DNS registries to predict malicious intent well before a domain name becomes operational called Premadoma. The paper contrasts this approach with reactive measures like blacklists, which only offer protection after some harm has already been done. Premadoma works by leveraging recent insights into the ecosystem of malicious domain registrations, focusing explicitly on facilitators employed for bulk registration and similarity patterns in registrant information. It has been successfully deployed in the production environment of the ".eu ccTLD" registry to detect and prevent malicious registrations, and has contributed to the take down of 58,966 registrations in 2018. The authors argue that Premadoma serves as a deterrent which substantially increases the cost for attackers and disincentivize malicious actors from launching campaigns. This is because Premadoma can prevent registrations from entering the zone file by predicting their

maliciousness. This means that attackers would have to go to greater lengths to register malicious domains, which would make it more difficult and expensive for them to do so.

An article based on DNS dataset for malicious domains detection by Claudio Marques et al. [28], focuses and provides a relation between a list of DNS datasets and methods used in classifying malicious and benign domain names. It begins by acquiring lists of already classified malicious and non malicious domains. With regards to Machine Learning (ML), the gave a better accuracy in identification, but happened to be limited to which parameters of the data set where used. When identified based on country codes or georaphical locations is seemed highly irrelevant as the even though the domain came from a particular location, it could not guarantee the same to be repeated. Using subdomains as a parameter gave sufficient results but failed to hit the exact mark as the use of subdomains for malicious purposes was limited. The study gives an extensive insight into which parameter one would look for which would provide efficient ways to identify malicious and non malicious domains. The data provides the identification and valuation of two classes of domains, malicious and non malicious which is valuable to computer and data science investigations. The detection of malicious domains is a critical challenge in the fight against cybercrime. Recent studies have shown that DNS data can be used to identify malicious domains with a high degree of accuracy. By developing and deploying more sophisticated detection systems, we can better protect our online environment from malicious attacks.

The registration phase, marking the inception of newly registered domains, holds potential for identifying potentially malicious domains. In their study on detecting malicious and abusive domain names, Kidmose et al. [29], examined two distinct features associated with the registration process to comprehend domain name behavior. The first feature, Pre-registration, scrutinizes the period preceding the initial update to the zone. During this phase, it can be ensured that the domain has not been misused on the Internet, as it has not yet been published in the TLD zone. Pre-registration encompasses the 2LD name, the registrant's payment information, billing address, and physical address. This data is furnished to the registrar to facilitate information submission to the registry during zone updates, including authoritative name server (ANS) details for the 2LD and registrar identification. The second feature, Post-registration (Pot reg), delineates the timeframe following the zone update, during which the 2LD becomes resolvable and its information accessible. At this juncture, the registry can analyze sample queries to the domain and adjust the TLD and ANS correspondingly. Post-registration is further subdivided into Pre-abuse and Post-abuse periods, with the point of domain involvement in malicious activity demarcating the transition between these sub-features. While this distinction aids in establishing a substantial timeline for the transition of a domain from benign to malicious, unequivocally determining whether a domain resolution is malicious or benign remains challenging. The paper explores when abuse detection can occur in a domain's life-cycle, which entities are capable of detecting abuse, the detection features employed, and the real-world application aspects, outlining avenues for future research and enhancements. Considering the uncertainties associated with detection timing, leveraging post-registration data could to some extent support the analysis of pre-registration. The paper underscores the deficiency in identifying potential malicious domains during the registration process, attributed to legal constraints hindering registrars from sharing information or the redundancy of registrars diminishing the incentive to detect malicious domains at this stage. It emphasizes the untapped potential for registrars or registries to detect malicious domains through innovative registrant-based features.

Utilizing visual representations of data facilitates precise and clear identification of patterns and behaviors. Despite extensive research and application of graph databases across disciplines, their utilization for network analysis remains limited. In their study titled "A Graph Database-Based Approach to Analyze Network Log Files", Lars Diederichsen et al. [30], explore the potential of em-

ploying a graph database for real-time log file analysis within a Network Security Monitoring (NSM) environment. They aim to integrate information from diverse sources to identify relationships between various network traffic entities in real-time logging NSM environments. Commencing with the collection of log files from Zeek, a prominent Intrusion Detection System, 'conn.log', 'dns.log', and 'http.log' are acquired. Rigorous monitoring ensures no log data is overlooked, and proper log rotation is maintained. Once prepared for parsing, logs are converted into Python dictionaries for subsequent processing. These dictionaries serve as parameters for a function that generates nodes and relationships from extracted log data, modeling a data graph imported into a Neo4j graph database. Utilizing the Py2neo2 Python library, nodes and relationships are established, and graph data is imported into Neo4j. Analysis of dns.log reveals logged information such as timestamp, IP addresses, ports, protocols, queried domain, query answer, and metadata. Evaluation identifies four entity types to use as nodes: DNS representing the connection, IP representing host computers, Host representing queried domain names, and Connection representing overall connections linking graphs. Interconnecting nodes yield at least six relationships. The "Connection" node establishes a directed connection to the "DNS" node via the "CONTAINS" relationship. The "IP" node is linked to the "DNS" node through the "HAS_DNS_REQUEST" relationship. Both "IP" and "DNS" nodes share a propertyless relationship, "HAS_QUERY," directed to the "Host" node representing the queried domain. The "RESOLVED_TO" relationship connects the "DNS" node with "Host" or "IP" nodes resulting from DNS query answers, incorporating time-to-live values and timestamps as properties. This enables easy querying of the graph for host resolution time or domain associations. Employing a graph database enables seamless display of log data and simplifies the identification of referral sequences with single queries. This facilitates the reconstruction of malicious cyber activities and simplifies querying for related hosts, thereby reducing analysis time in such cases.

The study, titled "A Graph Database-Based Method for Network Log File Analysis", presents a novel approach to analyzing network log files using a graph database. Led by K. Sharma et al. [31], the paper emphasizes the importance of continuously monitoring log files for immediate processing and analysis, with results then imported into the graph database. Operating within a Network Security Monitoring (NSM) environment, the research focuses on collecting DNS logs from Zeek IDS and extracting valuable insights. Utilizing a Degraph cluster shard, the system predicates and replicates predicates across the cluster, allowing queries to be executed on any node while handling joins over distributed data. Dgraph Zero serves as the central component, managing the cluster and coordinating database operations and analysis, while Dgraph Alpha nodes, equipped with indexed data, handle processing tasks. At least one Zero and Alpha node each are required to manage stored data effectively, enabling the cluster to store and process massive amounts of data without compromising research speed. The indexing process generates binary files containing the indexed data, significantly reducing processing time whenever data is refreshed. To store the indexed data and finalize the analysis, Neo4j, a graph database with a query language tailored for describing both graph structure and queries, is utilized. The study demonstrates the potential of real-time log record evaluation in NSM contexts using a graph database, albeit with untested scalability beyond networks of 45 members. Recommendations include enhancing code generation for monitoring larger networks and adding more log files, such as those generated by Zeek, to bolster investigative capabilities. Additionally, the performance of clustering multiple instances of Neo4j, particularly integrating data from various Zeek sensors monitoring different networks, warrants further investigation. Query efficiency experimentation is conducted using data from Neo4j's official sandbox, which, while well-structured, is limited in volume. This dataset encompasses networking and IT administration information with specific characteristics. The research also involves exporting data from Neo4j to CSV files and importing them into MySQL and MongoDB databases for further

analysis.

Chapter 4

Implementation

This section delves into the methodology employed in gathering and analyzing data for malicious domain name detection over the span of four months. Our primary objective is to proactively identify and monitor newly registered domain names, observing their day-to-day activities. By studying the behavior of domain names before they are blacklisted, we aim to deduce their malicious profiles. Our approach combines various established techniques for detecting malicious domain names. We meticulously select the most significant DNS-based and domain name-based features from existing research. The goal is to transform DNS data into a graph representation, facilitating the identification of relationships between domain names belonging to the same malicious campaign. Ultimately, we aim to detect suspicious domains by identifying their most salient features.

4.1 Experimental method

Every day, countless Newly Registered Domains (NRD's) are added to the vast landscape of the internet, courtesy of various registrars worldwide. To kickstart our experiment as depicted in figure 4.1, we delve into the realm of these NRD's by harnessing data from the whois database. This indispensable tool meticulously monitors the internet and facilitates access to a staggering array of domains, spanning over 100,000 Top-Level Domains (TLD's), offering valuable insights such as domain owner details, contact information, and addresses [32].

With the aid of WhoisDB, we gain entry into this trove of newly registered domains, which we have meticulously constrained to a daily sample size of 1,000 domains. Employing a series of meticulously crafted scripts, our project unfolds systematically. Initially, a script is deployed to harvest NRDs, followed by a randomization process to ensure a fair selection, limiting the collection to the prescribed 1,000 domains per day over the following five days.

Subsequently, another program takes the helm, tasked with querying resource records for the chosen 1,000 NRDs. It is essential to note that during this process, certain domain records may be concealed or entirely absent. In light of this, we have opted to disregard such domains, as they fail to provide the requisite depth of insight required for our experiment's scope. At the same time, the Whois lookup script that can be seen in listing A.5 was collecting WhoIS records from the randomly chosen 1,000 domains.

Additionally, after extracting the IP addresses from the resource records, we employ the python script in listing A.3 to scan these IPs across the AbuseIPDB, Greynoise, and FireHOL databases, facilitating the identification of malicious IPs. Simultaneously, we conduct domain scans using the Spamhaus API to uncover any malicious domains.

Finally, from the combined list of malicious IPs and domains, we analyze common patterns and behaviors, which are then graphically represented using Neo4j. In addition, the entropy for random generated domain names was calculated.

Through this intricate process, we aim to unravel the dynamics of the ever-evolving landscape of Newly Registered Domains, shedding light on their characteristics and behaviors within the digital ecosystem.

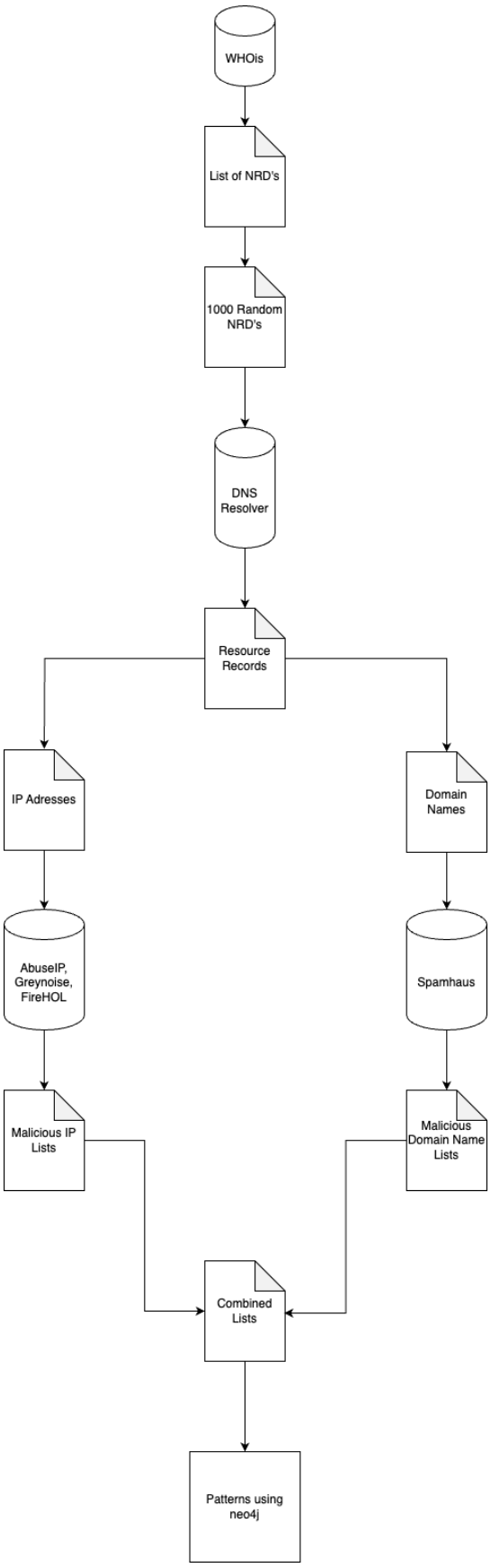


Figure 4.1: Implementation Flow

4.2 DNS based Features

Based on our research in the literature review, for the 1,000 daily collected NRD's we query resource records from the DNS servers using DNS resolvers. The resource records we prioritise are the 'A', 'AAAA', 'SOA', 'MX', 'NS', 'TXT'. The selected RR's allow to look into the different behavioural patterns that can be made visible, which in turn guides into understanding the malicious intend of the newly registered domains.

4.2.1 A Record

Indeed, beginning with an 'A' record or IPv4 address is paramount when navigating the intricate web of the internet and discerning the ownership of a domain. By analyzing 'A' records, we can discern whether they have been exploited in nefarious schemes or previously linked to suspicious behavior, thus enhancing cybersecurity measures and safeguarding internet users. While most websites maintain a single 'A' record, some prominent platforms adopt multiple 'A' records, employing a technique known as round robin load balancing. By distributing traffic across multiple servers helps improve scalability and reliability. This can enhance the performance of websites, applications, and services by preventing any single server from becoming overwhelmed with requests.

Like a coin has two sides, malicious actors can abuse round-robin DNS to redirect traffic to unauthorized servers under their control. In this experiment 'A' record leads the way that binds the newly registered domain names and their legitness. Obtained from various blocklists which are covered in the blocklist section of the paper allows into looking which domains are classified as malicious from birth and as to why they are classified as such. This had lead us to observe various patterns such as a single IP hosting multiple NRD's on a given day. Behaviours where NRD's contained and changed IP address over a given span of five days.

4.2.2 AAAA Records

DNS 'AAAA' records match a domain name to an IPv6 address. DNS 'AAAA' records are exactly like DNS 'A' records, except that they store a domain's IPv6 address instead of its IPv4 address. The experiment looks at the 'AAAA' records in the absence of the 'A' records. IPv6 can be used in major forms of attacks like 'ARP' spoofing, 'DOS' attacks and even malware distribution. As the 'AAAA' record is till in the beginning of its journey attacks using such are on a lower scale as to attacks and misuse of the IPv4. As nearly or not encountered many malicious identified domains do not contain IPv6 records.

4.2.3 SOA Records

Identifying the entity responsible for a domain is an arduous task, bordering on the impossible. Malicious actors frequently exploit Whois privacy to obfuscate their identities, often providing false registration information. The threat research conducted by "FireEye" on Iranian operations underscores the critical importance of such threat intelligence which lead to identify the actors behind cyber incidents of all kinds — including fake news campaigns and outright election tampering [33].

DNS 'SOA' (Start of Authority) records are invaluable in uncovering behavioral patterns of malicious domains, enriching Cyber Threat Intelligence (CTI) efforts. By examining various attributes within these records, analysts can discern subtle clues that hint at nefarious activities [34].

Considering the Administrative Contact Information provided in 'SOA' records, malicious domain owners often conceal their identities using obfuscated or anonymized contact details. By scrutinizing variations or inconsistencies in contact information across multiple malicious domains, the experiment aims to uncover patterns indicating potential linkage or shared ownership. For example, discovering similar or identical contact information associated with multiple malicious domains could signify a coordinated effort by threat actors. This pattern is usually created due to cost in terms of time and money. Furthermore, the Serial Number within 'SOA' records unveils insight to another pattern where number updates with each modification to the zone data, enables one to track changes over time. Malicious actors may employ domain fluxing techniques, evidenced by erratic serial number updates across iterations of 'SOA' records for the same domain. For instance, if a domain associated with a malware campaign exhibits frequent and irregular serial number changes, it may suggest ongoing attempts at evasion or manipulation.

In combination with 'NS', changes to the authoritative Name Servers specified in 'SOA' records can reveal behaviours of malicious intent. Attackers may attempt to manipulate these servers to redirect traffic or obscure their activities. Monitoring alterations in name server information within 'SOA' records enables to detect suspicious patterns, such as frequent changes or deviations from standard practices. For example, if a domain undergoes sudden and unexplained shifts in its authoritative name servers, it could indicate a hijacking attempt or unauthorized modifications.

4.2.4 NS Records

The nameserver record, as spoken in the background section indicates which DNS server is authoritative for that domain i.e. which server contains the actual DNS records. While NS records themselves do not directly detect malicious behavior, they can indirectly help in identifying certain types of malicious activities or mis-configurations.

By projecting on such malicious activities brings to light patterns including, like Short Lifespans, Frequent Changes, Suspicious Nameservers. Behaviours majorly observed during the duration of the experiment are frequent changes and suspicious behaviour. This allowed us to determine whether a NRD has 'NS' records that are previously related to malicious activities. We further looked into multiple different NRD's that contained the same 'NS' records that were linked to other listed malicious domains.

4.2.5 TXT Records

'TXT' (Text) records in the (DNS) are typically used to store arbitrary text data associated with a domain. 'TXT' records, inherently designed to detect malicious behavior provided valuable metadata and context when looking for patterns of a potential malicious domain. We were able to observe that most NRD's that were potentially flagged as malicious domains contained no 'TXT' records or 'TXT' records with misconfigured DNSSEC.

Misconfigured records for the malicious domains were indicated by have only SPF Version 1 and no 'DMARC', or 'DKIM'. Furthermore these records failed to have any signs of domain verification or ownership confirmation. The 'TXT' records also highlighted which domains were related to the same NS records of the other malicious domains.

4.2.6 MX Records

A DNS 'Mail Exchange' (MX) record directs email to a mail server. The 'MX' record indicates how email messages should be routed in accordance with the Simple Mail Transfer Protocol (SMTP, the standard protocol for all email). Like 'CNAME' records, an 'MX' record must always point to another domain.

4.3 Whois Records

Whois server as an internet record listing that identifies who is the registrant of a specific domain. A Whois record encompass comprehensive details about the owner of a domain name including their name, contact number, email, country of registration, and key dates like the most recent update and expiration as specified by ICANN. The purpose of whois records is to bolster cybersecurity efforts by revealing the individuals or entities behind domains, enabling appropriate actions against malicious activities. Moreover, these records aid companies in implementing fraud prevention measures [35].

This study involved collecting Whois records to determine the geographical distribution of malicious domains using ASN data. Additionally, it aimed to differentiate domains with the same IP registered on the same day but belonging to different registrants, and those not associated with the same cloud infrastructure.

4.4 Data Representation

To facilitate easy manipulation of the data, a json representation was created. As seen below for each domain there is a records table that contains all the resource records along with the date of their collection. So, for each day the records table was filled with the new date and the new resource records until it reaches the five days of observation.

```
{
 "example.com": {
    "records": [
      {
        "date": "yyyy-mm-dd",
        "A": [
          "xxx.xxx.xxx.xxx"
        ],
        "AAAA": [
          "xxxx:xxxx:xxxx::xxxx"
        ],
        "MX": [
          "mailhost1.example.com"
        ],
        "NS": [
          "ns1.exampleserver.com"
        ],
        "TXT": [
          "\"v=spf1 include:spf.xxxxx.xxxxxxxxxx-xxxxxxxxxxxx.com ~all\""
        ],
        "SOA": [
          "ns1.exampleserver.com. admin.example.com. 1111111111 86400 7200 4000000 11200"
        ]
      }
    ]
  }
}
```

4.5 Blocklists

Domains may find themselves on blocklists due to a variety of reasons, such as involvement in spamming, malware dissemination, phishing, botnet operations, or other illicit activities. These

blocklists are typically maintained by email service providers, anti-spam organizations, blacklist aggregators, and other entities focused on internet security. By flagging blocklisted domains, these measures aim to prevent users from accessing potentially harmful sites, thereby reducing the risks associated with malicious online activities. Such blocklists serve as essential protective barriers, shielding users from interactions with potentially dangerous domains. Moreover, there exist blacklist aggregators that compile data from diverse sources, including user reports, malware analyses, and automated detection systems, to create comprehensive blocklists.

When a domain is placed on a blocklist, access to it may be restricted or entirely blocked by internet service providers, email servers, web browsers, and other relevant platforms. This proactive approach plays a crucial role in mitigating the dangers associated with malicious domains, enhancing users' privacy, security, and overall internet experience. Prominent blocklists commonly used for improving internet security include AbuseIPDB, Greynoise, Spamhaus, FireHOL, among others. These resources provide blocklists or threat intelligence feeds that can be integrated into various security tools. The selection of the most suitable option depends on specific requirements; for instance, Greynoise or AbuseIPDB may be preferred for broader threat intelligence and investigation purposes, while Spamhaus's Domain Blocklist may be ideal for email spam filtering, and FireHOL IP lists for IP threat blocking. The level of detail provided about identified threats varies across these resources. Information can be combined from multiple resources to get a more comprehensive picture of potential threats.

As a best choice and based of integrity of information provided we have used the following blocklist to compare and create the needed behaviours and patterns:

- **AbuseIPDB:** Functions as a collaborative threat intelligence platform. It is a community-driven database where users can submit information about malicious IP addresses, domains, URL's, email addresses, and other threat indicators. Relies on user submissions and contributions from the security community. Provides details about reported threats, including timestamps, types (IP, domain, URL), and reporting users (optional). AbuseIPDB being the leading source of obtaining malicious IPs leverages us to discover potentially malicious domains that may contain already blocklisted IPs [36].

- **Greynoise:** Focuses on identifying malicious IP addresses and domains associated with various cyber threats. By analyzing internet traffic patterns through automated traffic analysis techniques to collect data, it categorizes IPs and domains based on their observed behavior. We are able to use the threat categorization (e.g. malware, spam) and reputation scores for identified IPs and domains that are offered by greynoise. This allows us to view deeper into the investigation of malicious patterns of NRD's and prioritise which pattern would suit best to determine behaviours [37].

- **Spamhaus:** Primarily targets email spam. It is a blocklist of domains known to be used for sending spam emails. Employs a combination of automated analysis and manual investigation which allows us to identify spam sources. Spamhaus provides lists of domains that are categorized as spam sources. As mentioned in the background section of this paper, not all records can be queried for all domains, this has been countered to some extent though Spamhaus. Spamhaus' analysis and investigative approach allows us to obtain knowledge on the maliciousness of a NRD that has no or hidden records. Though 'MX' records we are able to relate spam emails originating from listed domains [38].

- **FireHOL:** Allowing us to correlate and provide extra validation that the malicious domain lists provided by the above three blocklists are majorly reported. FireHOL provides lists of IP

addresses associated with various malicious activities, including spam, malware distribution, botnets, and Denial-of-Service (DoS) attacks. Gather data from various sources, including security researchers, spam filtering services, and network operators. Contain IP addresses categorized by threat type (spam, malware, etc.). Some providers might offer additional details like timestamps or associated domains. Network administrators can use them to configure firewalls and intrusion detection/prevention systems (IDS/IPS) to block traffic from malicious IP addresses [39].

4.6 Analysis Pattern

Analysis of the gathered data unveils discernible patterns, serve as crucial indicators for understanding the diverse behaviors exhibited by NRD's, distinguishing between malicious and benign activities. These patterns not only facilitate a deeper understanding but also provide invaluable insights into the registration, usage, and lifecycle of domains, enabling enhanced detection and mitigation strategies against cyber threats. By scrutinizing these patterns, we are able to decipher subtle nuances that characterize legitimate domain activities from those indicative of malicious intent. Such analysis encompasses various factors including registration frequency, domain age, naming conventions, hosting infrastructure, and network traffic behavior of which are gathered from the records mentioned in the features subsection of this section. The identification of these overarching patterns sets the stage for a more granular examination of the distinct six analysis patterns inherent in malicious behaviors of NRD's.

The following patterns were selected from the 6 primarily formed patterns which promised better insight into the NRD's behaviour be it malicious or benign.

1. **Domains with same malicious IP's:** Known widely, there is no strict limit on the number of 'A' records a domain possesses lead us to observe and form a pattern at NRD's who shared the same malicious IP's. Although this feature being very beneficial leads into major misuse. This pattern helps identify which domains use IP fluxing and allows us to understand whether a domain resolves to multiple IP addresses evade detection.

2. **Domains with same malicious IP's registered on the same day:** As mentioned above a single domain can contain multiple IP addresses. Registering multiple domains on the same day and have them point to the same set of malicious IP addresses benefits malicious actors in a gracious way. Forms of cyber-crime such as phishing campaigns, malware distribution and DDoS are assisted by exploiting this feature. This pattern allow us to see behaviours of domains that are registered with multiple IP's at their creation. It further allows us to view into the lifetime of the domains and whether the IP's in question points to another domain.

3. **Single day domain records:** Evading identification being one of the major goals of the cyber-criminals world, allows them to facilitate attacks simultaneously. A common practise is making use of DNS by creating new domains just for a day and then cease their existence, which allows us to form patterns that help us understand the behaviours exhibited by such domains. This pattern is commonly seen in DDoS attacks.

4. **Domain with IP Changes:** Seen from the above three patterns IP fluxing is a common strategy used by malicious actors. Combining the fore-mentioned pattern allows us to differentiate which domains would exist for just one day and which domains would change the 'A' records the very next day of its birth or after a given time period.

5. **DNSSEC enabled email authentication:** As discussed, DNSSEC being a security measure, is not a regulatory feature that needs to be in effect when creating a domain. The pattern allowed us to observe behaviours on how malicious domains use DNSSEC and to what extent.

6. **Entropy for random character domain names:** It is a very common practice for adversaries to create random character domain names to perform potentially malicious acts. Through observation in the data it was discovered that a lot of malicious domains were randomly generated. Calculating the entropy of these domains gave an understanding of how random and unpredictable they were, aiding in identifying potentially malicious activities more effectively.

4.7 Graphical Representation using NEO4j

Neo4j, a robust database renowned for its high performance in handling graph data, employs a user-friendly query language alongside ACID transactions. This framework enables us to navigate a dynamic network structure of nodes and relationships rather than static tables. Leveraging its 'ACID' properties - Atomicity, Consistency, Isolation, and Durability - empowers us to construct and explore intricate patterns utilizing the interconnections among five key features. Meaning that [40]:

- **Atomicity:** If a transaction fails the database remains unchanged.
- **Consistency:** Each transaction ensures the database remains in a consistent state upon completion.
- **Isolation:** Multiple transactions that happened at the same time cannot affect each other.
- **Durability:** Committed transactions can be recovered.

Neo4j uses a query language called cypher. Cypher is capable of querying and modifying data so it is suitable for property graphs. The syntax of cypher language is simple and similar to SQL language where the queries are structured using several clauses. Some of the most common clauses are [41]:

- **MATCH:** Defines the specific patterns to be queried within the database.
- **WHERE:** Adds constrains to the pattern in MATCH clause.
- **RETURN:** Defines what data will be returned.
- **SET:** Updates the labels and properties on nodes and relationships.
- **CREATE:** Creates nodes and relationships.
- **MERGE:** Creates a pattern in the graph if it is not already exist.

With its scalability, Neo4j facilitates the examination of behavioral trends gleaned from data spanning a period of four months. Utilizing Neo4j, we efficiently isolate and discern patterns through various scripts. One such script, illustrated below, serves as the foundation for generating the initial pattern discussed in section 4.6.

```
import json
from neo4j import GraphDatabase

# Neo4j connection parameters
uri = ""                # Your Neo4j URI
username = ""           # Your Neo4j username
password = ""           # Your Neo4j password

# Function to import DNS data from JSON object
def import_dns_data(tx, dns_data):
    for domain, records in dns_data.items():
        for record in records['records']:
            domain_name = domain
            date = record['date']
            # Extract all record types
            a_records = record.get('A', [])
            aaaa_records = record.get('AAAA', [])
            mx_records = record.get('MX', [])
            ns_records = record.get('NS', [])
            txt_records = record.get('TXT', [])
            soa_records = record.get('SOA', [])
            # Create or merge domain node
            tx.run("MERGE (d:Domain {name: $domain_name})",
                   domain_name=domain_name)
            # Create relationships for A records
            for ip_address in a_records:
                tx.run("MERGE (i:IPAddress {address: $ip_address})",
                       ip_address=ip_address)
                tx.run("MATCH (d:Domain {name: $domain_name}), "
                       "(i:IPAddress {address: $ip_address}) "
                       "MERGE (d)-[:RESOLVES_TO]->(i)",
                       domain_name=domain_name, ip_address=ip_address)
            # Create relationships for AAAA records
            for ipv6_address in aaaa_records:
                tx.run("MERGE (i:IPAddress {address: $ipv6_address})",
                       ipv6_address=ipv6_address)
                tx.run("MATCH (d:Domain {name: $domain_name}), "
                       "(i:IPAddress {address: $ipv6_address}) "
                       "MERGE (d)-[:RESOLVES_TO]->(i)",
                       domain_name=domain_name, ipv6_address=ipv6_address)
            # Create relationships for MX records
            for mx_record in mx_records:
                tx.run("MERGE (m:MailServer {name: $mx_record})",
                       mx_record=mx_record)
                tx.run("MATCH (d:Domain {name: $domain_name}), "
                       "(m:MailServer {name: $mx_record}) "
                       "MERGE (d)-[:USES_MAIL_SERVER]->(m)",
                       domain_name=domain_name, mx_record=mx_record)
            # Create relationships for NS records
            for ns_record in ns_records:
                tx.run("MERGE (n:NameServer {name: $ns_record})",
                       ns_record=ns_record)
                tx.run("MATCH (d:Domain {name: $domain_name}), "
                       "(n:NameServer {name: $ns_record}) "
                       "MERGE (d)-[:USES_NAME_SERVER]->(n)",
                       domain_name=domain_name, ns_record=ns_record)
            # Create relationships for TXT records
            for txt_record in txt_records:
                tx.run("MERGE (t:TextRecord {value: $txt_record})",
                       txt_record=txt_record)
                tx.run("MATCH (d:Domain {name: $domain_name}), "
                       "(t:TextRecord {value: $txt_record}) "
                       "MERGE (d)-[:HAS_TEXT_RECORD]->(t)",
                       domain_name=domain_name, txt_record=txt_record)
            # Create relationships for SOA records
            for soa_record in soa_records:
                tx.run("MERGE (s:SOARecord {value: $soa_record})",
                       soa_record=soa_record)
                tx.run("MATCH (d:Domain {name: $domain_name}), "
                       "(s:SOARecord {value: $soa_record}) "
                       "MERGE (d)-[:HAS_SOA_RECORD]->(s)",
                       domain_name=domain_name, soa_record=soa_record)

# Establish Neo4j connection and run the import transaction
driver = GraphDatabase.driver(uri, auth=(username, password))
with driver.session() as session:
    # Read JSON data from file
    with open('/home/user/Desktop/DNS/mal.json', 'r') as file:
        dns_data = json.load(file)
    # Import DNS data
    session.write_transaction(import_dns_data, dns_data)

    # Create relationships between domains with the same IP address
    session.run("MATCH (i:IPAddress)<-[:RESOLVES_TO]-(d:Domain) "
                "WITH i, collect(d) AS domains "
                "WHERE size(domains) > 1 "
                "UNWIND domains AS d1 "
                "UNWIND domains AS d2 "
                "WITH d1, d2 "
                "WHERE id(d1) < id(d2) "
                "MERGE (d1)-[:SHARES_IP_WITH]->(d2)")

# Close Neo4j driver
driver.close()
```

This script extracts data from JSON-formatted files, creating or merging domain nodes and subsequently associating all RR's with their respective domain names. Following the script, we establish links between domains sharing the same IP addresses, as shown in figure 4.3. While the initial nodes are established through a uniform procedure, the relationships between them diverge based on distinct behaviors observed during the analysis.

Neo4j can be used as a desktop client running through localhost to create the graphs in a very user friendly environment that you can export the results either in 'csv' or 'png' format. The UI of Neo4j can be seen in figure 4.2.

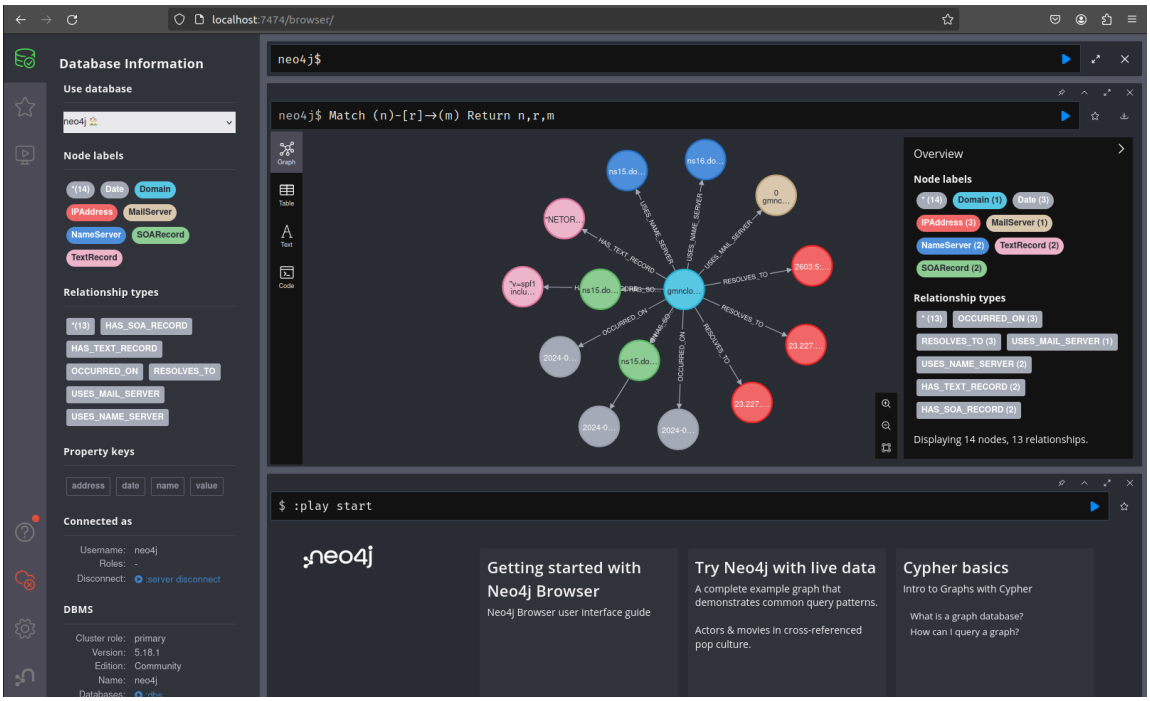


Figure 4.2: Neo4j user interface

The Neo4j graph model used for the patterns can be seen in figure 4.3. The relationships and nodes were selected based on how useful the information would be to explain the different patterns and behaviour of malicious domains. The relationships are described in details in section 4.7.1.

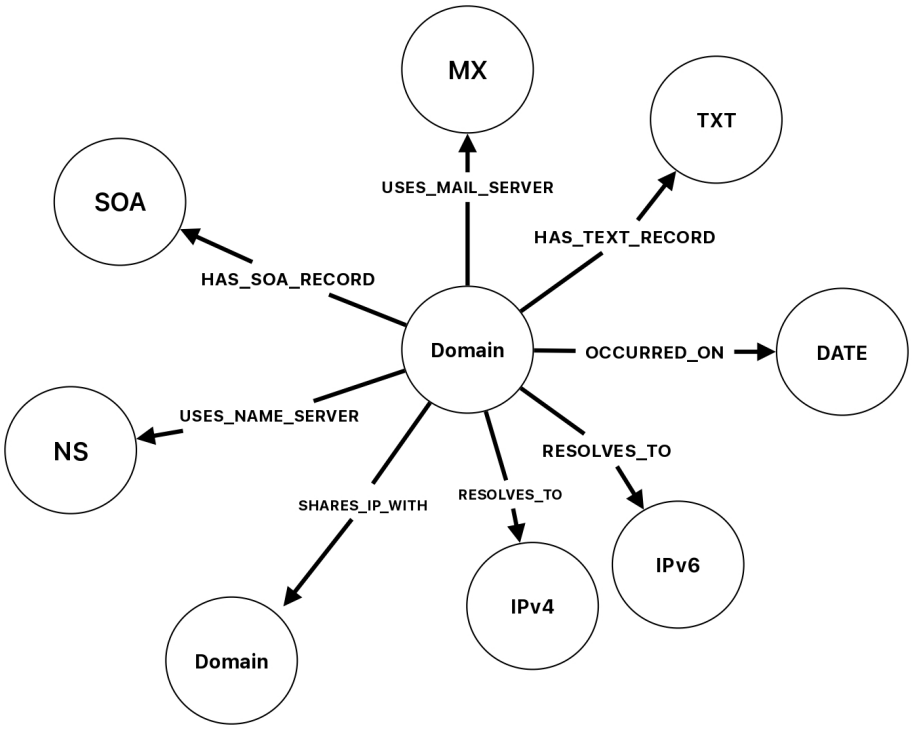


Figure 4.3: The neo4j graph model

4.7.1 Relationships

Neo4j graphs contain nodes and relationships. In order for the graph to be understandable each relationship connects specific nodes. The relationships created for the results can be seen in table 4.1 give a better view of the patterns and how the DNS Resource Records are connected with the domains.

- **OCCURRED_ON:** (Domain to Date relationship) Each domain node is related to a date node through the 'OCCURRED_ON' relationship. This relationship indicates when the DNS data associated with the domain were collected.
- **RESOLVES_TO:** (Domain to IP Address relationship) Each domain node is related to one or more IPv4 (A) or IPv6 (AAAA) address nodes through the 'RESOLVES_TO' relationship. This relationship indicates which IP addresses are associated with the domain.
- **USES_NAME_SERVER:** (Domain to Name Server relationship) Each domain node is related to one or more name server nodes through the 'USES_NAME_SERVER' relationship. This relationship indicates which name servers are used by the domain.
- **HAS_TEXT_RECORD:** (Domain to Text Record relationship) Each domain node is related to one or more text record nodes through the 'HAS_TEXT_RECORD' relationship. This relationship indicates which text records are associated with the domain.

- **HAS_SOA_RECORD:** (Domain to SOA Record relationship) Each domain node is related to one or more SOA record nodes through the 'HAS_SOA_RECORD' relationship. This relationship indicates which SOA records are associated with the domain.
- **USES_MAIL_SERVER:** (Domain to Mail Server relationship) Each domain node is related to one or more mail server nodes through the 'USES_MAIL_SERVER' relationship. This relationship indicates which mail servers are used by the domain.
- **SHARES_IP_WITH:** (Domain to Domain relationship) If two domains share the same IP address, they are related to each other through the 'SHARES_IP_WITH' relationship. This relationship is established based on domains having the same IP address.

These relationships help represent the connections between different entities in the DNS data, allowing for querying and analysis in the Neo4j database.

Relationship	**From -> To**	**Node Color**
OCCURRED_ON	Domain -> Date	grey
RESOLVES_TO	Domain -> IP	red
USES_NAME_SERVER	Domain -> NS	blue
HAS_TEXT_RECORD	Domain -> TXT	pink
HAS_SOA_RECORD	Domain -> SOA	green
USES_MAIL_SERVER	Domain -> MX	beige
SHARES_IP_WITH	Domain -> Domain	

Table 4.1: Relationships in the graphs

Chapter 5

Results

This section presents the findings of the study, examining the data collected over a period of four months from 1st of December 2023 to 31st of March 2024. The analysis encompasses the total collection of 120,000 NRD which focuses on the observation of their resource records and the collection of 23,999 malicious domains and categorizing them into patterns. The study yielded several key findings that shed light on the behaviour of malicious domains. These findings are presented and analyzed below, contributing to a deeper understanding of DNS security. The results are organized based on our own observation and research review of similar studies. This approach facilitates a systematic exploration of the data and allows for a comprehensive evaluation of the study outcomes.

5.1 Patterns And Behavior of Malicious Domains

5.1.1 Domains With Same Malicious IPs

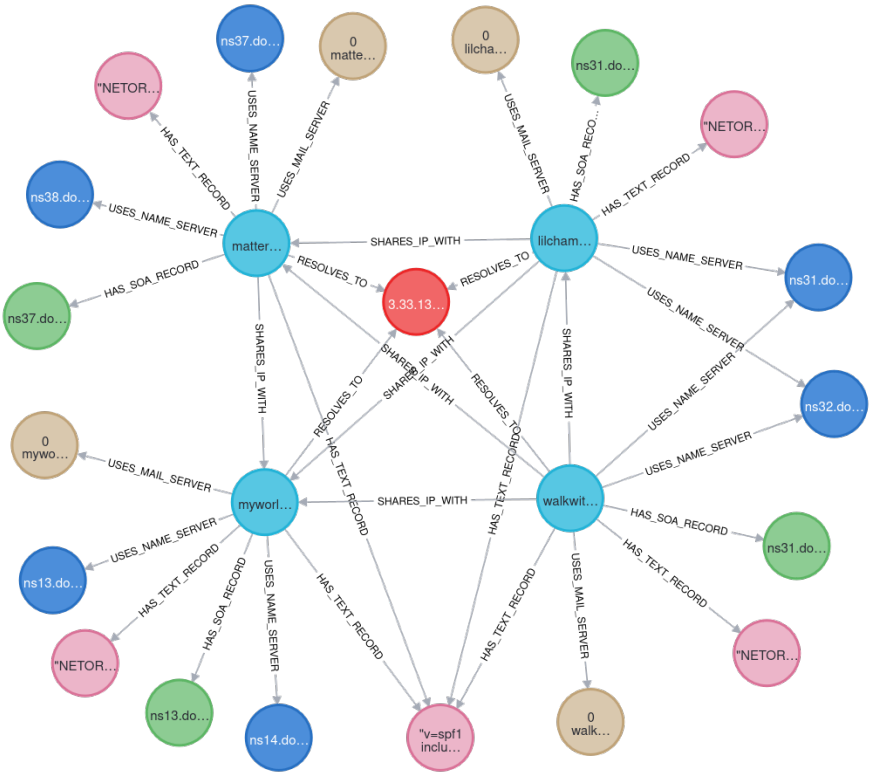
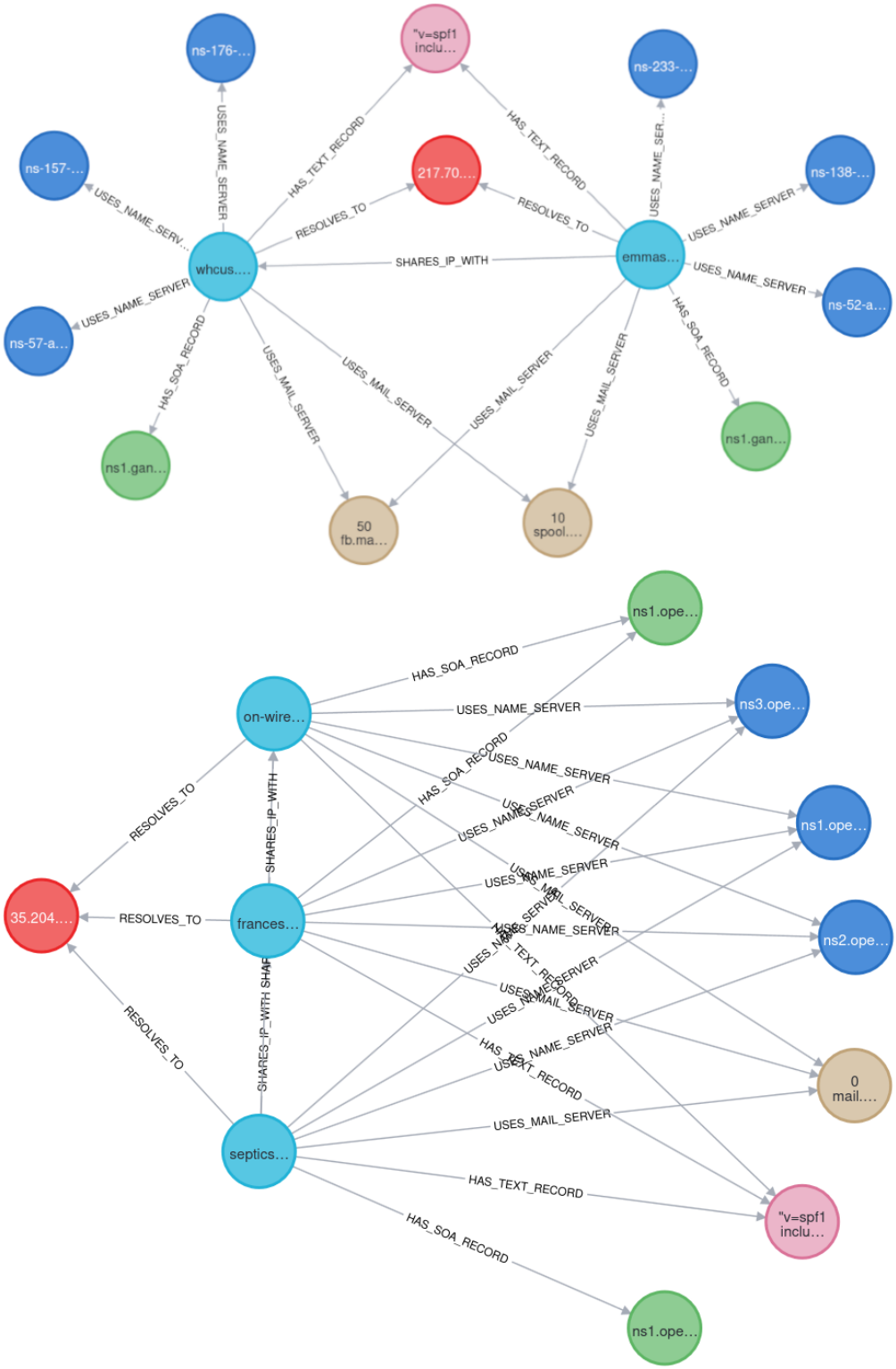


Figure 5.1: Four domains with the same IP

As mentioned in the previous section, the pattern where domains have the same malicious IP provide us with different behaviours. Looking at over 24,000 malicious domain names we primarily

observed the commonalities shown by domain names with respect to the resource records. The domains showed to having same 'SOA' records, 'NS' records, 'MX' records, and same 'TXT' records as depicted in figures 5.1 and 5.2. The number of observed domains with the same malicious IP was 4,999.



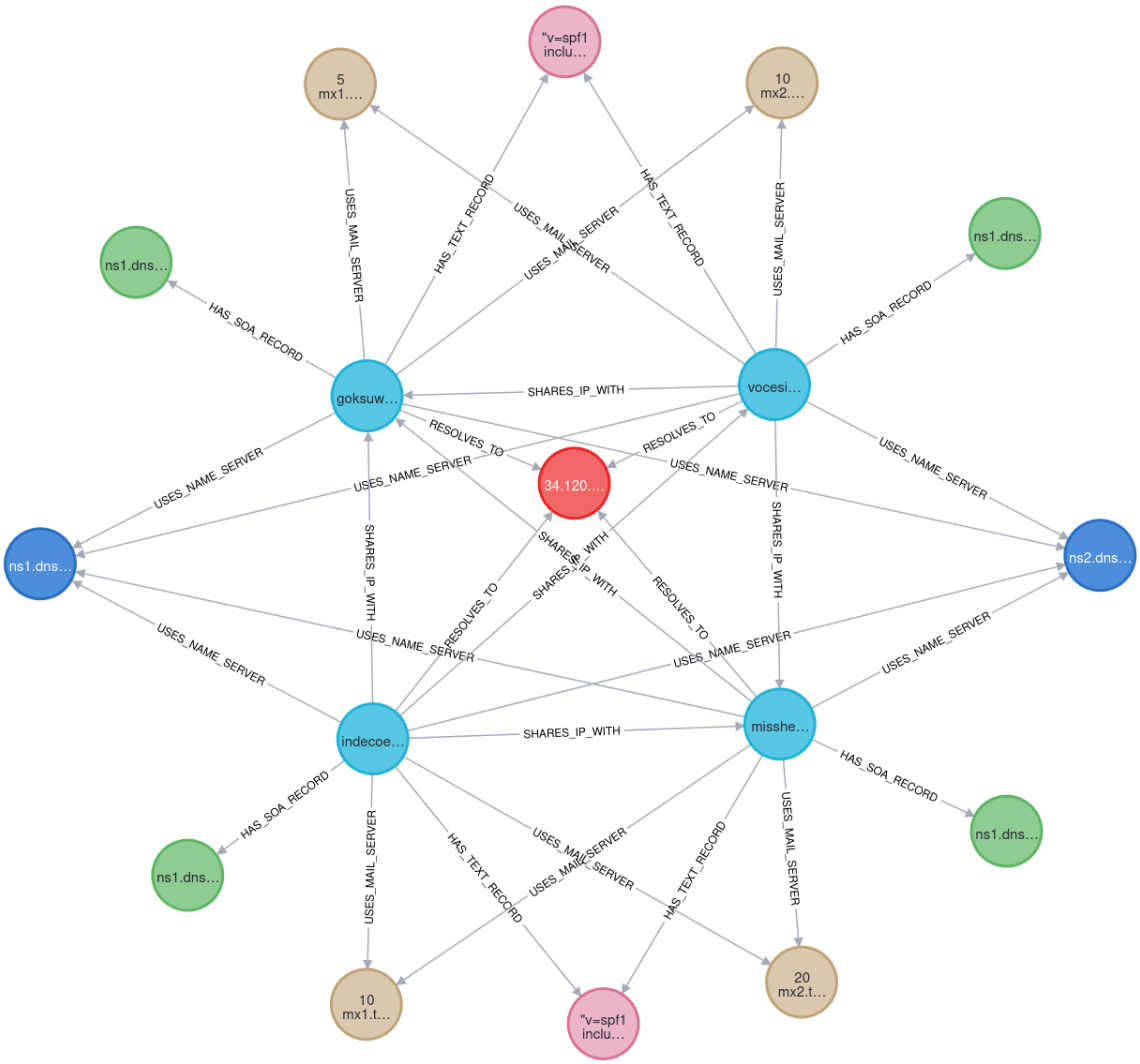


Figure 5.2: Domains with the same IP for three different IP addresses

5.1.2 Domains With Same Malicious IPs Registered On The Same Day

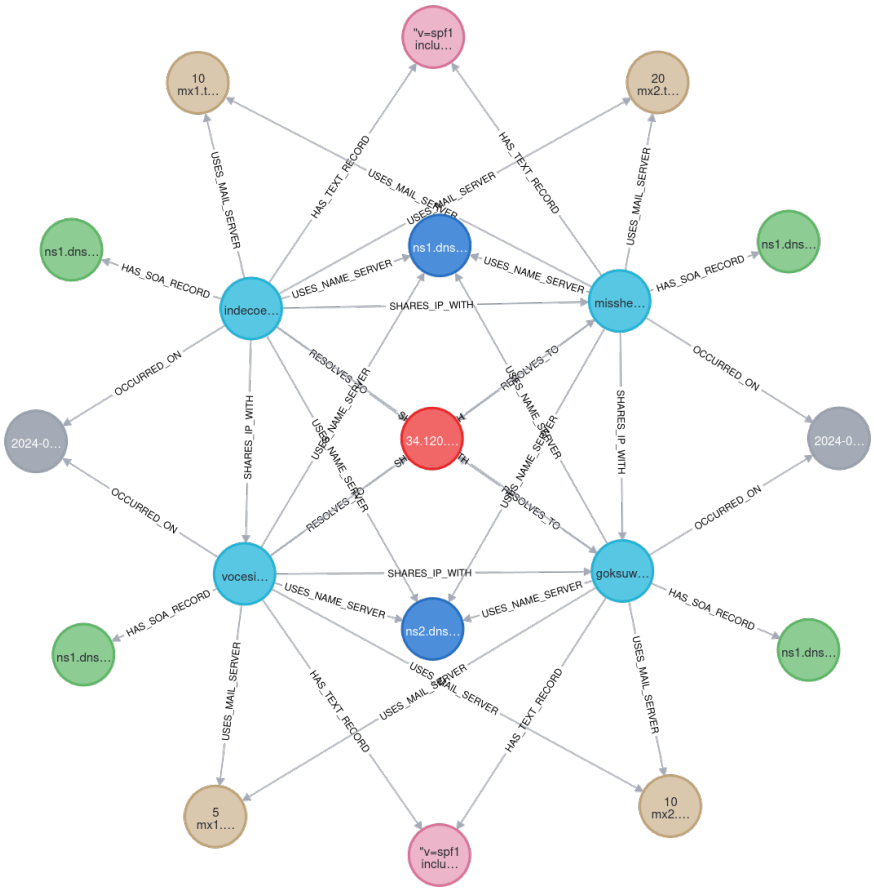


Figure 5.3: A domain with the same IP for the first two days

Understanding that by registering domains with the same malicious IP's on the same day, provides attackers with operational efficiency, redundancy, and enhanced capabilities for evading detection and executing complex, large-scale malicious activities. Figure 5.3 shows the relationship between four different domains and their commonalities over a period of two days where two domains on either side been registered on the same day. Visualising a larger picture figure 5.4 allows us to view for nine different NRD, for a four different days. The number of observed domains with the same malicious IP's registered on the same day was 1,799.

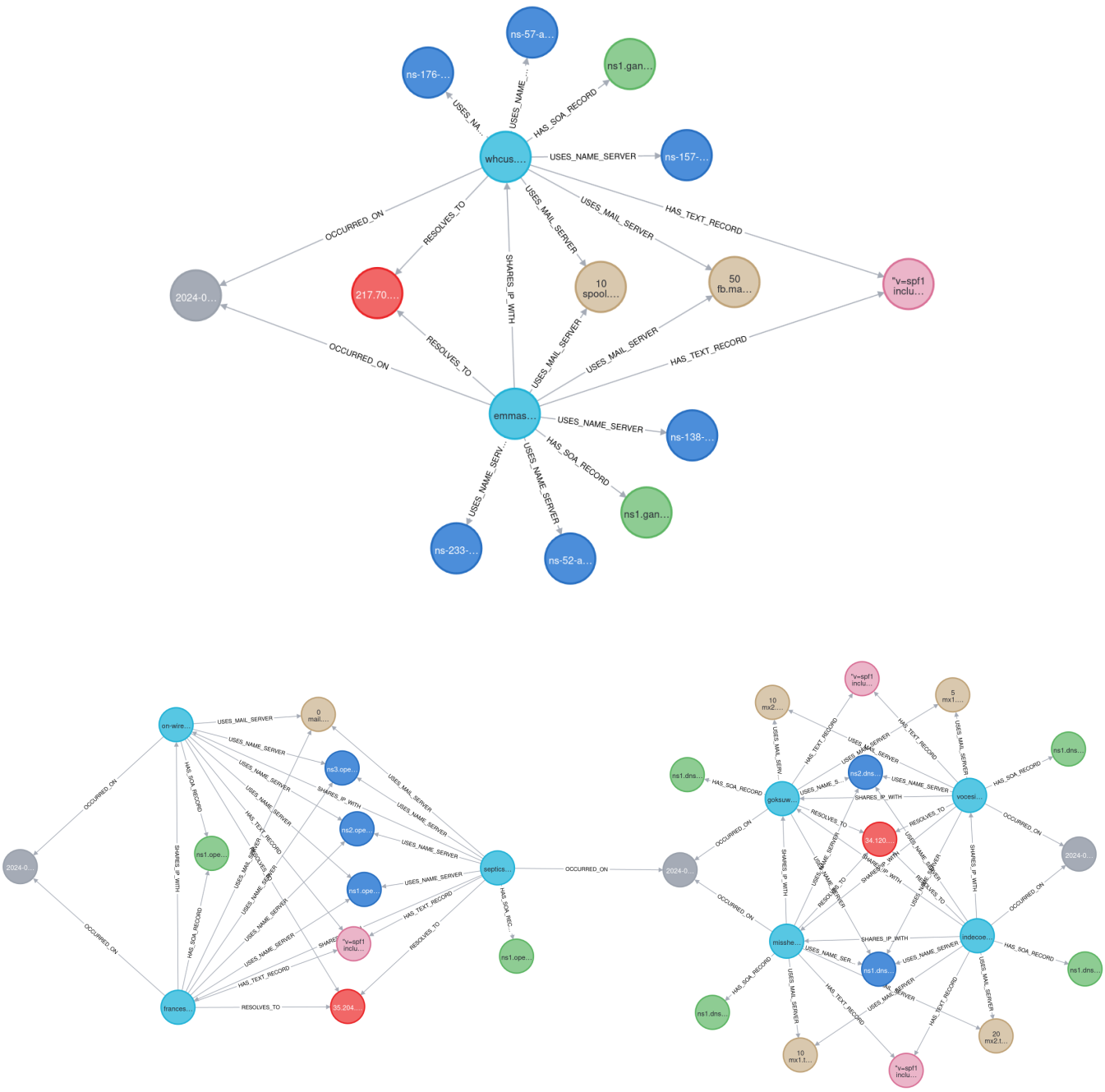


Figure 5.4: The same domain with a changed IP for the next four days

5.1.3 Single Day Domain records

As mentioned before it is a common practise for attackers to register domains and keep them alive for only one day. As observed from the data for 1,999 domains it was discovered that there were domains with records only for one day. Furthermore, as can been seen in figure 5.5 there were domains with the same IP but they do not have any other common resource records.

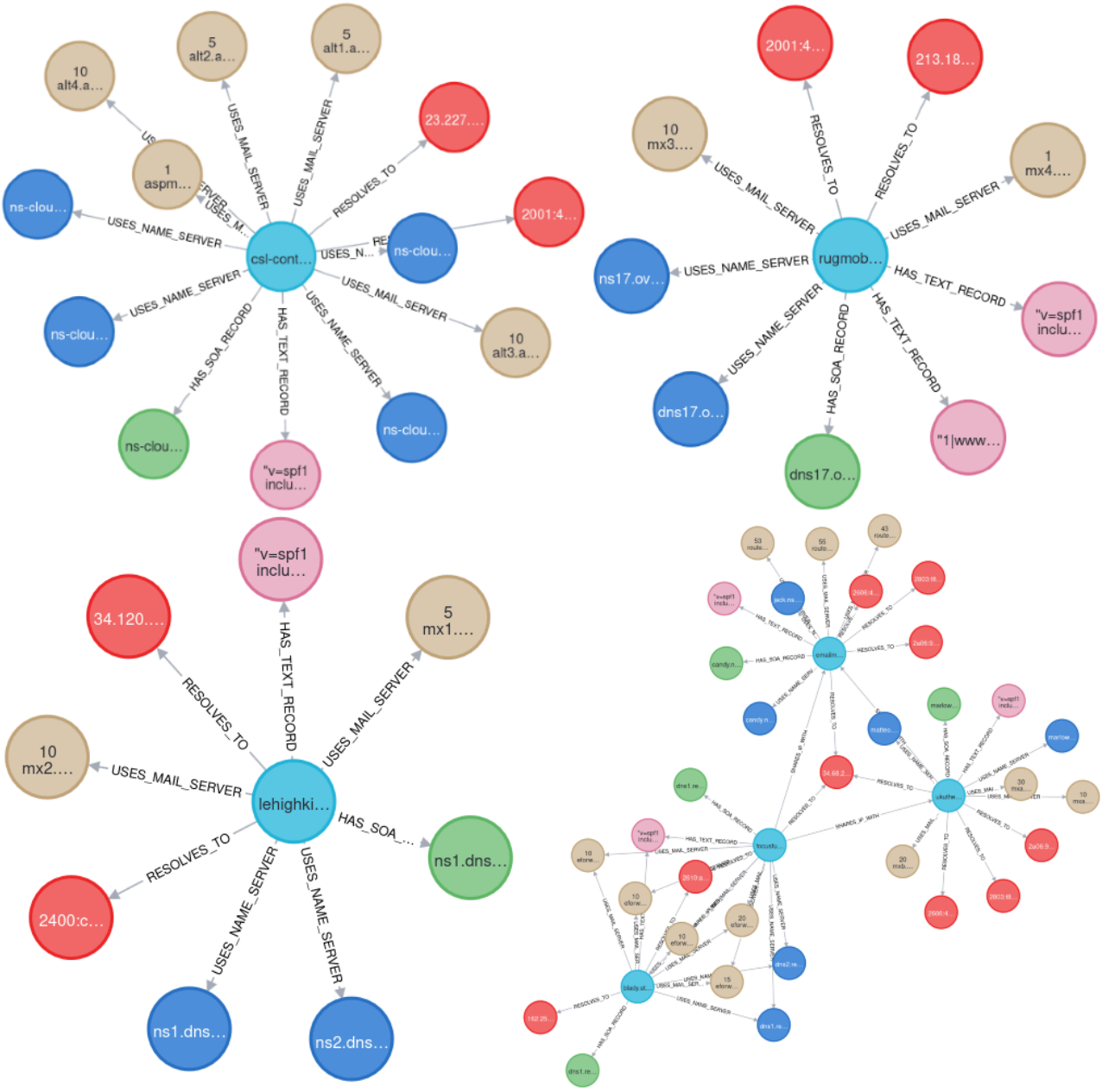


Figure 5.5: Domains with records for only one day

5.1.4 Domain with IP Changes

As written before it is a common strategy for the adversaries to register a domain with a specific IP and after some time they changed it to a whole new IP. Observing 499 domains with IP changes whose reference can been seen in figure 5.6 the domain has the same IP the first two days and the third day it completely changed its IP and kept it for the next two days (figure 5.7). All the other resource records remained the same for the whole five days.

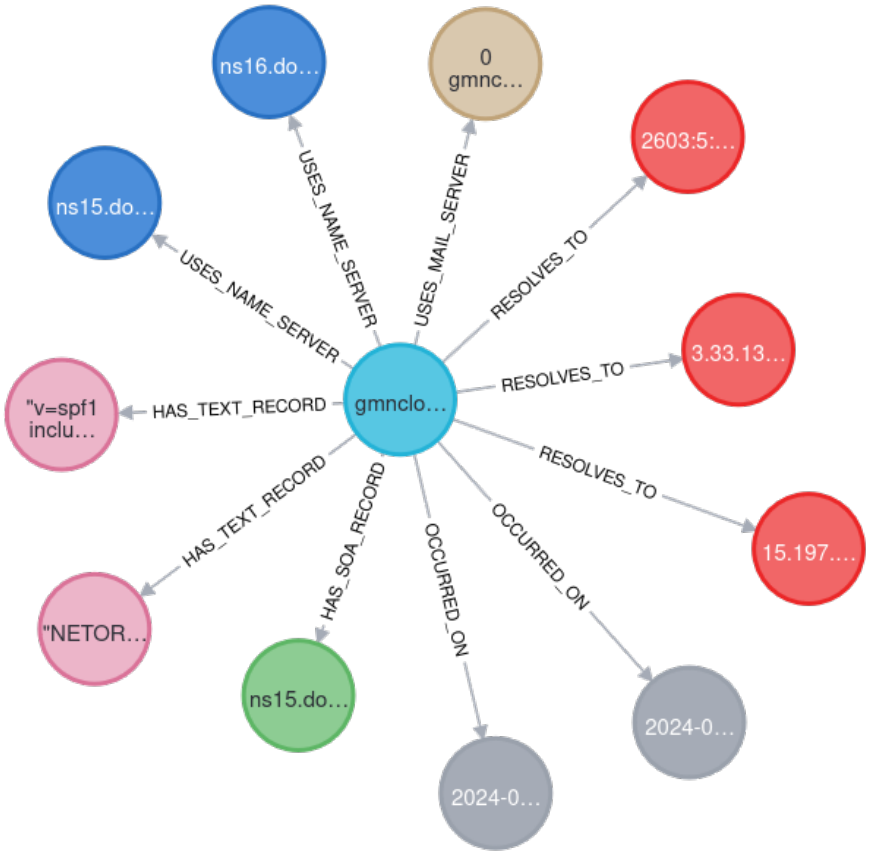


Figure 5.6: A domain with the same IP for the first 2 days

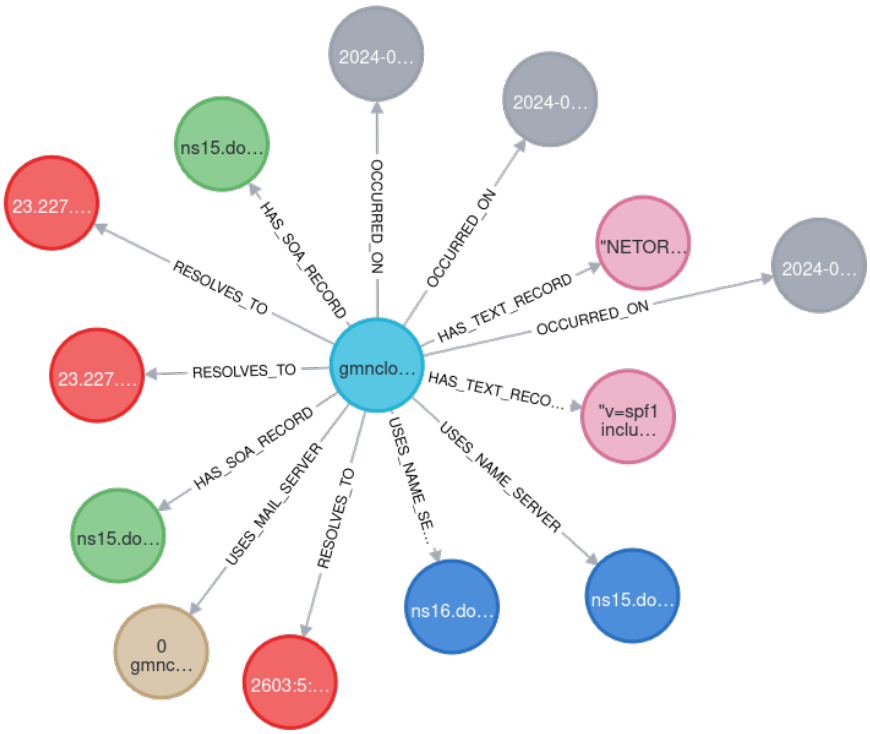


Figure 5.7: The same domain with a changed IP for the next 3 days

5.1.5 DNSSEC Enabled email authentication:

As described in Chapter 4, the Spamhaus API was used to inspect potentially malicious domains. Upon reviewing the Spamhaus results, it was observed that a significant portion of domains flagged as malicious had no resource records throughout the entire five-day data collection period. Consequently, no additional information was available for these domains. Further investigation into the reasons behind their flagging revealed that none of them had DNSSEC enabled or had DNSSEC enabled with basic email authentication, indicated by "v=spf1". Thus, as observed through Spamhaus, these domains were considered illegitimate and likely malicious.

In figure 5.8 it is evident that the 8% of the domains that where collected had the DNSSEC disabled. Furthermore, 20% of them had the DNSSEC enabled meaning that not all the signatures such us DKIM and DMARC where enabled. Although, DNSSEC was enabled to 72% of the domains it is important to note that not all of these domains were legitimate.

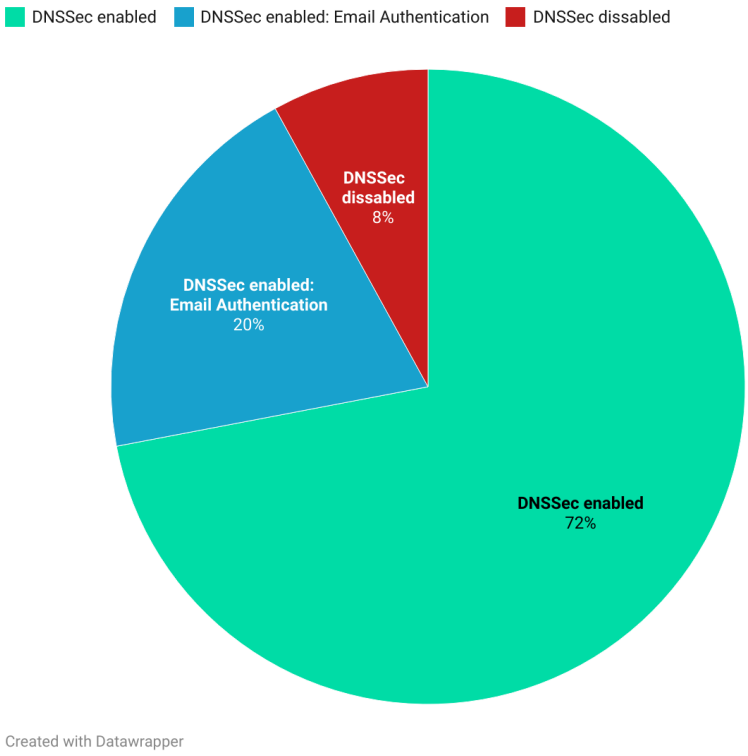


Figure 5.8: DNSSec enabled

5.1.6 Entropy For Random Character Domain Names

As discussed in Chapter 4, attackers commonly generate domain names using randomly generated characters. This pattern was quite frequently observed in the datasets. The calculated entropy was based on a mathematical concept in information technology which determines the frequency of each character (letters, digits, hyphens) appearing in a string. So, domain names which are short and/or have large numbers of repeated characters will show low entropy, while domain names which are longer and/or contain large numbers of distinct characters will portrait high entropy. This includes only the main part of the domain without the TLD label. Using a Python script, detailed in the listing A.6, we calculated the entropy of these domain names. Table 5.1 displays domain names

with low entropy, while table 5.2 lists those with high entropy. Figure 5.9 shows the distribution of entropy across the whole database.

TLD	Entropy
8.rest	0.00
ff.work	0.00
99.fo	0.00
ttttttttttttttttttttttttttttttttttop.top	0.37
77777767.xyz	0.54
a22222.cc	0.65
e0000.cc	0.72
e2222.cc	0.72
nnni.us	0.81
k555.tv	0.81
esee.store	0.81
7779.ws	0.81
fbbb.shop	0.81
qhhh.org	0.81

Table 5.1: Low entropy domain names

TLD	Entropy
xn–ihq5mwmu0nk6ev7ntsan2nba626jyxa270bl38ai45d1uplkb.com	4.71
wwaj3naqokorfcw2schbungcea8572xfearx8v4jzt.monster	4.55
bamu174x1ne0pe8z9j4yrzh1gjvh2d.online	4.42
bestluxurysuv2023mexico854695.life	4.35
cheap-family-vacations-us-5246075.zone	4.32
cinematography-course-71364.bond	4.31
cloudcomputingsearch175261.life	4.29
truck-driver-jobs-intl-5318984.xyz	4.28
xn–ockvfya9734aouk6n1euvr8vq.com	4.28
uhakmoytw3pqjdeskmbcin.top	4.28
51d8tnpzmk16a3lnrgc9208.com	4.26
window-replacement-jobs-49596.bond	4.25
cinematography-course-67497.bond	4.24
xn–zf0bm3j5pizqj9ta48dxz3b.com	4.21

Table 5.2: High entropy domain names

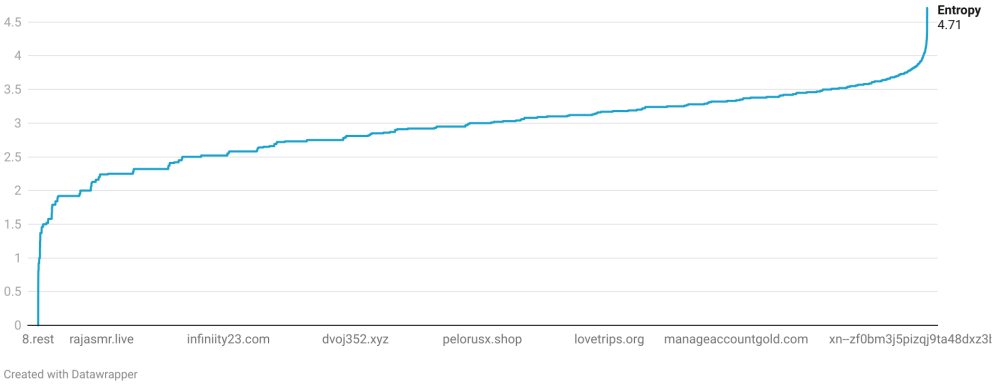


Figure 5.9: Distribution of entropy

Chapter 6

Discussion & Evaluation

Outlining the paper structure, this chapter includes a discussion about the findings for each of the research questions based on the results in Chapter 5.

Building on the patterns and behaviors identified in the results, this section critically evaluates the findings using various criteria from the related works section. This collective understanding provides a basis for answering the research questions.

Focusing on the RR's, the patterns showed significant commonalities among various NRD's. A primary focus was on 'A' records, which represent the IP addresses of domains. Many NRD's shared common 'A' records. This behavior highlighted how many IP addresses flagged as malicious by the three blocklist databases were registering new domains daily. This insight led to questions such as how many days new domain names are registered by individual IP addresses, how long these domains exist, and what actions they perform. Analysis showed that more than half of the total NRD's were identified as malicious. Many of these NRD's were registered almost daily by the same IP addresses. Observing these NRD's over a span of five days revealed that most NRD's remained active for the entire five days, whereas less than half of the observed malicious NRD's had records for fewer days, with the majority existing for just one day. This outcome provided insight into why certain domains existed for more or fewer days. Using blocklists as analytical tools, it was found that many domains with shorter lifespans were related to phishing attacks, with the remainder associated with spam and phishing.

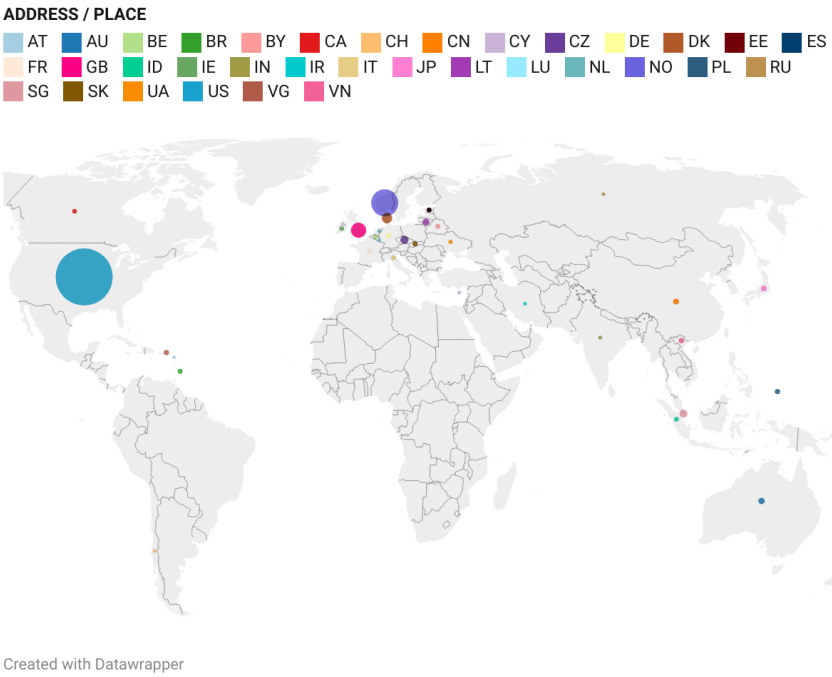


Figure 6.1: ASN Map

Following the information obtained from the records query, we mapped out the characteristics

and behaviors of these domains. As detailed in Chapter 4, 'SOA' records provide authoritative information about a DNS zone, including the primary name server for the zone, the responsible party's email address, the serial number (to track changes), refresh interval, retry interval, expiration time, and minimum 'TTL' for cached information. Using the 'RNAME' field from 'SOA' records, we identified the domain owners by cross-referencing with readable and complete Whois records accredited by ICANN.

This approach enabled us to determine which ASN and Registrar country codes the domains predominantly originated from. Figure 6.1, based on harvested Whois records, shows the global distribution of ASNs associated with malicious IP addresses. A significant number of these malicious IP addresses were originated from the US and Europe, providing insight into the geographic emergence of malicious IP's.

Examining registrar details in Figure 6.2, we observed that the majority of malicious NRD's were registered through US-based registrars, followed by European and Asian countries. This analysis showed that many malicious NRD's shared common 'SOA' records. A notable feature was the use of the 'MNAME' field, which specifies the domain's primary name server, allowing verification of the domain against its records.

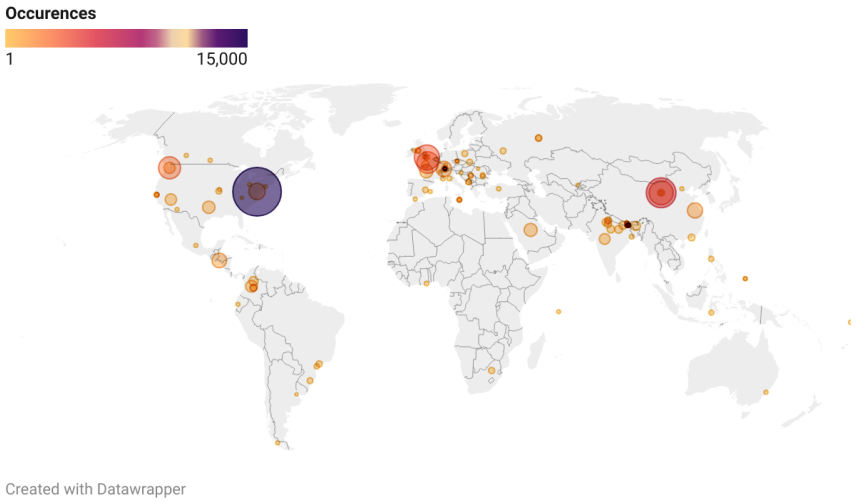


Figure 6.2: Registrar Map

Verification using the 'MNAME' field in 'SOA' records, as mentioned in Chapter 4, indicated which DNS server is authoritative for a domain, i.e., which server contains the actual DNS records. This helped in identifying the authoritative DNS server. Comparing the patterns revealed that most malicious NRD's shared common 'NS' records. This indicated that malicious 'NS' records, or those previously flagged as malicious, were not being thoroughly checked. This observation further supported our analysis of the origins of these 'NS' records on a global scale. We found that most malicious domains exhibited frequent changes in their records over a span of five days. Domains with records of three days or less showed patterns of short lifespans.

Additionally, the analysis of 'MX' and 'TXT' records provided insight into the role of DNSSEC in securing DNS. DNSSEC use was examined from two perspectives: domain validity based on the Spamhaus Blocklist for domains without records or with redacted records, and whether email authentication of DNSSEC was enabled for domains with 'TXT' records. Data analysis through Spamhaus indicated that 8% of NRD's had DNSSEC disabled. Spamhaus, which primarily targets email spam, provided these insights. Observations from Figure 5.8 in Section 5 showed that around

20% of the domains had DNSSEC enabled with basic email authentication, indicated by "v=spf1" which does not guarantee legitimacy.

Common 'NS' records further revealed patterns in NRD's containing randomly generated characters, which were characterized as randomly generated domains. Using a Python script, entropy for the domains was calculated. Figures 5.1 and 5.2 depict entropy for domains with high and low entropy, respectively. Figure 5.9 provides a clearer distribution of entropy for randomly generated domains across the database. Although these domains were identified, they often lacked detailed records. Spamhaus provided further understanding of why these domains were flagged as malicious.

Based on the above analysis, we move to answer the questions presented in Chapter 1. These questions form the core of our thesis experiment.

To begin with the first question, "What are the patterns and behaviors of malicious domains?" The answer is found in the patterns and behaviors detailed in Chapter 5. Easily and not easily identifiable patterns from Whois data, such as domain information about registrants, highlight the need for registrars to implement better algorithms to prevent the registration of already or potentially malicious domains. The association factors can be closely checked by analyzing existing 'SOA' and 'NS' records and their relationships.

The next question, "What are the common characteristics of malicious domains?" is primarily answered through the behavior of 'A' records. Most classified malicious domains typically share the same 'A' records and similar RR's. Common characteristics also include the registration of domains on the same day with identical and additional 'A' records. The patterns discovered also revealed significant IP fluxing, where domains change 'A' records frequently while leaving other records unaltered. Finally, many malicious domains have records for a single day before ceasing to exist, often related to phishing and spam attacks identified by blocklists.

The third question, "How do these patterns and characteristics provide us with clues for identifying a potentially malicious domain even prior to its creation through a Newly-Registered Domain?" can be answered by combining insights from the first two questions. Understanding the patterns, behaviors, and common characteristics of malicious domains allows for the observation and validation of an NRD to classify it as malicious or benign. For example, 'A'records or IP addresses can help registrars determine if the address under which a domain will be registered is benign or malicious. Additionally, having datasets to differentiate non-malicious and potentially malicious domains would be a good strategy for registrars. Similarly, 'SOA' and 'NS' records provide registrars with the knowledge needed to authenticate the 'NS' records of previously existing domains to validate their benignity. As mentioned previously, DNSSEC is recommended but not required, supporting the findings presented in the DNSSEC pattern in Chapter 5. It is also noted that even though domains have DNSSEC enabled, they often only meet the basic level of security.

Visual representations of the gathered data using Neo4j provide an answer to the fourth and final question: "How does Neo4j enhance the understanding of these patterns?" Neo4j, known for its robust capabilities, enabled the navigation and analysis of a dynamic network structure comprising nodes and relationships. This allowed for the identification of patterns within the data. By leveraging Neo4j's ACID properties, the database ensured that observations of data and relationships remained consistent and reliable throughout the extensive data collection period of four months. Utilizing Cypher, Neo4j's powerful query language, facilitated a deeper understanding and visualization of the correlations between domains and their RR's. Cypher's expressive syntax made it easier to perform complex queries, and present the data in a meaningful way. As a result, the insights gained from these visualizations contributed significantly to the comprehension of patterns and relationships within the dataset, demonstrating the value of Neo4j in handling and interpreting large-scale, interconnected data.

This discussion of the implementation and results illustrates how post-registration information can inform better checks and practices by registrars, enabling them to identify potentially malicious domains even before their creation.

Chapter 7

Conclusion

This study provides several key insights into the behavior of Newly Registered Domains and their association with malicious activities. By systematically examining the patterns and behaviors linked to malicious domains, we have identified common characteristics that can help flag potentially harmful domain names.

The paper begins by emphasizing the crucial role of the Domain Name System in internet infrastructure, translating human-readable domain names into IP addresses. This central role also makes DNS a prime target for malicious actors. DNS attacks are on the rise, causing significant financial damage and increasing complexity in the threat landscape. Malicious domains are frequently used for phishing, spam, and other cyber attacks, underscoring the importance of understanding their behaviors and characteristics.

Previous research has analyzed the behaviors of malicious domains, focusing on various DNS features and patterns. However, gaps remain in the comprehensive understanding of these behaviors, particularly in relation to NRD's. This study aims to bridge these gaps by providing a detailed analysis of the patterns and characteristics of malicious domains.

The research employed a robust experimental method, collecting and analyzing data from NRD's over an extended period. Various DNS-based features, such as 'A' records, 'AAAA' records, 'SOA' records, 'NS' records, 'TXT' records, and 'MX' records, were examined over a span of five days. Whois records were also analyzed to gather additional information on the domains. Data representation and analysis were conducted using tools like Neo4j for graphical representation, allowing for the identification of patterns and relationships among the domains.

The study identified several key patterns and behaviors associated with malicious domains. These include domains sharing the same malicious IP's, domains registered on the same day, single-day domain records, frequent IP changes, and the presence of DNSSEC. Additionally, entropy analysis of domains with randomly generated names provided further insights into their malicious nature. The results highlighted that malicious domains often share common traits, such as shared 'A' records, frequent IP changes, and short-lived existence, which are typically associated with phishing and spam attacks.

As we reflect on the results, it is also important to acknowledge the limitations of this study, which suggest avenues for future research.

7.1 Limitations

Conducting experiments, especially those involving intricate methodologies and advanced technologies, inevitably presents various challenges. Throughout our research, we encountered several difficulties that impacted the progress and outcomes of our experiments. Addressing these challenges was crucial for ensuring the validity and reliability of our findings. Below, we detail the key difficulties faced and the strategies employed to mitigate their effects.

7.1.1 Sample Preparation

Determining the appropriate format for collecting and storing data turned out to be more complex than expected. Initially, we experimented with three different JSON formats, each influenced by studies cited in the literature review. However, these formats proved insufficient for effectively displaying the data. After finalizing the data format, we encountered another issue: several Python libraries for DNS were partially deprecated.

7.1.2 Data Producibility and Collection

Once the JSON format was finalized and the scripts were adjusted accordingly, we encountered limitations with the DNS resolvers. The experiment required a significant volume of data, but the number of query requests for the required RR was restricted. To mitigate this, we sent requests to various resolvers instead of relying solely on the closest one. While this approach simplified the querying process, it still restricted the number of records that could be queried for a single record. Specifically, we faced difficulties in collecting 'CNAME' records. Separate queries were needed to obtain CNAME's, and despite this effort, only 1% of the total collected NRD's included CNAME's. Consequently, we decided to exclude 'CNAME' records from the study entirely.

7.2 Future work

While this study focuses on the patterns and behaviors of malicious domains, there are several areas that could be further explored to enhance this research. One intriguing avenue is to investigate malicious IP's in greater depth, particularly examining whether attackers attempt to clear these IP's over time and how long such actions might take. Additionally, another valuable analysis would be to determine how long it takes for registrants to discover that a domain has been registered with an IP previously flagged as malicious multiple times during a significant period of time, and to understand the reasons behind the delays in addressing such issues.

In conclusion, this research makes a significant contribution to cybersecurity by introducing a novel approach to identifying potentially malicious domains through the analysis of NRD's. The systematic methodology, encompassing data collection, graphical representation, and detailed analysis offers a comprehensive understanding of the behaviors and characteristics of malicious domains. The insights gained from this study can inform improved practices and policies for domain registrars, ultimately enhancing online security. By identifying common patterns and traits of malicious domains, this research aids in the early detection and proactive mitigation of potential threats, thereby contributing to a safer digital landscape.

All used scripts for the collection and analysis of this thesis can be accessed through the following git repository [42].

Glossary

ASN Autonomous System Number. 22, 41

CNAME Canonical Name. 6, 10, 22, 45

CTO Chief Technology Officer. 3

DDoS Distributed Denial of Service. 1, 2, 10, 11, 24

DKIM DomainKeys Identified Mail. 21, 37

DMARC Domain-based Message Authentication, Reporting and Conformance. 21, 37

DNS Domain Name System. 1–10, 12–16, 18, 20–22, 24, 28, 29, 41, 44, 45

DNSSEC Domain Name System Security Extensions. 10, 12, 13, 21, 25, 37, 41, 42, 44

DoH DNS over HTTPS. 13

DoQ DNS over Quic. 13

DoT DNS over TLS. 13

IANA Internet Assigned Numbers Authority. 8

ICANN Internet Corporation for Assigned Names and Numbers. 6, 8, 10, 11, 22, 41

IP Internet Protocol. 5–10, 14, 16, 18, 20, 23, 24, 27, 29, 33–35, 40–42, 44, 45

ISP Internet Service Provider. 9

MX Mail Exchange. 6, 10, 20, 22, 23, 31, 41, 44

NRD Newly Register Domain. 3, 4, 10, 11, 18, 20, 21, 23, 24, 30, 33, 40–42, 44, 45

NS nameserver. 6, 20, 21, 31, 41, 42, 44

RR Resource Records. 5, 6, 20, 27, 40, 42, 45

SOA Start Of Authority. 20, 21, 29, 31, 41, 42, 44

SPF Sender Policy Framework. 21

SSL Secure Sockets Layer. 3

TCP Transmission Control Protocol. 1

TLD Top-Level Domain. 5, 8, 9, 15, 18

TTL Time To Live. 14, 41

TXT Text. 20, 21, 31, 41, 44

UDP User Datagram Protocol. 1, 2

URL Uniform Resource Locator. 23

Bibliography

[1] M. Anagnostopoulos, "Amplification dos attacks," in *Encyclopedia of Cryptography, Security and Privacy*, Springer, 2020, pp. 1–3.

[2] R. Fouchereau. "Idc 2023 global dns threat report." (Date accessed: 30/04/2024). (), [Online]. Available: https://efficientip.com/resources/cyber-threat-intelligence-idc-2023-global-dns-threat-report/.

[3] D. Mitchell. "How do i use the domain score to determine whether a domain is a threat?" (Date accessed: 01/05/2024). (), [Online]. Available: https://www.darkreading.com/threat-intelligence/how-do-i-use-the-domain-score-to-determine-if-a-domain-is-a-threat.

[4] *Domain names - concepts and facilities*, RFC 1034, Nov. 1987. DOI: 10.17487/RFC1034. [Online]. Available: https://www.rfc-editor.org/info/rfc1034.

[5] *What is dns*, https://www.cloudflare.com/en-gb/learning/dns/top-level-domain/, Accessed: 2023-10-30.

[6] *Tld*, https://www.cloudflare.com/en-gb/learning/dns/top-level-domain/, Accessed: 2023-10-30.

[7] *What is dns*, https://www.cloudflare.com/en-gb/learning/dns/dns-records/dns-ns-record/, Accessed: 2023-10-30.

[8] G. Schmid, "Thirty years of dns insecurity: Current issues and perspectives," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2429–2459, 2021. DOI: 10.1109/COMST.2021.3105741.

[9] *Dnslookup*, https://www.cloudflare.com/en-gb/learning/dns/what-is-dns/, Accessed: 2023-10-30.

[10] *Dnssec*, https://www.icann.org/resources/pages/dnssec-what-is-it-why-important-2019-03-05, Accessed: 2023-10-30.

[11] M. Anagnostopoulos, G. Kambourakis, E. Konstantinou, and S. Gritzalis, "Dnssec vs. dnscurve: A side-by-side comparison," in *Situational Awareness in Computer Network Defense: Principles, Methods and Applications*, IGI Global, 2012, pp. 201–220.

[12] *Dnssecurity*, https://www.cloudflare.com/en-gb/learning/dns/dns-security, Accessed: 2023-10-30.

[13] F. Personnel. "Gone phishing." (Date accessed: 01/05/2024). (), [Online]. Available: https://www.fbi.gov/news/stories/phishing-fraudster-sentenced-061319.

[14] D. Goodin. "Hackers take control of security firm's domain, steal secret data." (Date accessed: 01/05/2024). (), [Online]. Available: https://arstechnica.com/information-technology/2017/12/hackers-steal-security-firms-secret-data-in-brazen-domain-hijack/.

[15] *Malicious registed domain names*, https://www.paloaltonetworks.com/cyberpedia/what-are-malicious-newly-registered-domains, Accessed: 2023-10-30.

[16] *Icann*, https://www.icann.org/, Accessed: 2024-05-17.

[17] R.-V. Mahmoud, M. Anagnostopoulos, S. Pastrana, and J. M. Pedersen, "Redefining malware sandboxing: Enhancing analysis through sysmon and elk integration," *IEEE Access*, 2024.

[18] *The evolving dns threat landscape*, https://www.cloudflare.com/en-gb/learning/security/global-dns-hijacking-threat/, Accessed: 2024-05-17.

[19] D. Strom. "Typosquatting wave shows no signs of abating." (Date accessed: 17/05/2024). (), [Online]. Available: https://www.darkreading.com/threat-intelligence/typosquatting-wave-shows-no-signs-of-abating.

[20] P. P. Ltd. "Dnssec security: Unveiling potential attack vectors and mitigation strategies." (Date accessed: 17/05/2024). (), [Online]. Available: https://www.linkedin.com/pulse/dnssec-security-expert-unveiling-attack-vectors-mitigation/.

[21] M. Lyu, H. H. Gharakheili, and V. Sivaraman, "A survey on dns encryption: Current development, malware misuse, and inference techniques," *ACM Comput. Surv.*, vol. 55, no. 8, Dec. 2022. DOI: 10.1145/3547331. [Online]. Available: https://doi.org/10.1145/3547331.

[22] G. Kambourakis, M. Anagnostopoulos, W. Meng, and P. Zhou, *Botnets: Architectures, countermeasures, and challenges*. CRC Press, 2019.

[23] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "A survey on malicious domains detection through dns data analysis," *ACM Comput. Surv.*, vol. 51, no. 4, Jul. 2018. DOI: 10.1145/3191329. [Online]. Available: https://doi.org/10.1145/3191329.

[24] S. Hao, N. Feamster, and R. Pandrangi, "Monitoring the initial dns behavior of malicious domains," in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11, Berlin, Germany: Association for Computing Machinery, 2011, pp. 269–278. DOI: 10.1145/2068816.2068842. [Online]. Available: https://doi.org/10.1145/2068816.2068842.

[25] K. A. Messabi, M. Aldwairi, A. A. Yousif, A. Thoban, and F. Belqasmi, "Malware detection using dns records and domain name features," in *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, ser. ICFNDS '18, Amman, Jordan: Association for Computing Machinery, 2018. DOI: 10.1145/3231053.3231082. [Online]. Available: https://doi.org/10.1145/3231053.3231082.

[26] D. Yang, Z. Li, and G. Tyson, "A deep dive into dns query failures," in *Proceedings of the 2020 USENIX Conference on Usenix Annual Technical Conference*, ser. USENIX ATC'20, USA: USENIX Association, 2020.

[27] J. Spooren, T. Vissers, P. Janssen, W. Joosen, and L. Desmet, "Premadoma: An operational solution for dns registries to prevent malicious domain registrations," Dec. 2019, pp. 557–567. DOI: 10.1145/3359789.3359836.

[28] C. Marques, S. Malta, and J. P. Magalhães, "Dns dataset for malicious domains detection," *Data in Brief*, vol. 38, p. 107 342, 2021.

[29] E. Kidmose, E. Lansing, S. Brandbyge, and J. M. Pedersen, "Detection of malicious and abusive domain names," in *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, 2018, pp. 49–56. DOI: 10.1109/ICDIS.2018.00015.

[30] L. Diederichsen, K.-K. R. Choo, and N.-A. Le-Khac, "A graph database-based approach to analyze network log files," in *Network and System Security: 13th International Conference, NSS 2019, Sapporo, Japan, December 15–18, 2019, Proceedings 13*, Springer, 2019, pp. 53–73.

[31] K. Sharma and A. Kumar, "A graph database-based method for network log file analysis," in *2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART)*, 2022, pp. 545–550. DOI: 10.1109/SMART55829.2022.10047250.

[32] *What's in the whois*, https://www.whoisds.com/, Accessed: 2024-04-25.

[33] *Redefining critical infrastructure for the age of disinformation*, https://www.darkreading.com/threat-intelligence/redefining-critical-infrastructure-for-the-age-of-disinformation, Accessed: 2024-04-25.

[34] *Dns soa record*, https://www.cloudflare.com/en-gb/learning/dns/dns-records/dns-soa-record/, Accessed: 2024-04-25.

[35] W. D. Tools. "What is whois information and why is it valuable?" (Date accessed: 15/05/2024). (), [Online]. Available: https://www.domaintools.com/support/what-is-whois-information-and-why-is-it-valuable/.

[36] *What is abuseipdb*, https://www.abuseipdb.com/, Accessed: 2024-04-25.

[37] *Solving internet noise.* https://www.greynoise.io/, Accessed: 2024-04-25.

[38] *Who is spamhaus*, https://www.spamhaus.org/who-is-spamhaus/, Accessed: 2024-04-25.

[39] *About firehol*$_l$*evel*1, https://iplists.firehol.org/, Accessed: 2024-04-25.

[40] Neo4j. "Cypher and neo4j." (Date accessed: 14/05/2024). (), [Online]. Available: https://neo4j.com/docs/cypher-manual/current/introduction/cypher_neo4j/.

[41] Neo4j. "Clauses." (Date accessed: 14/05/2024). (), [Online]. Available: https://neo4j.com/docs/cypher-manual/current/clauses/.

[42] Chrysoula and Severen. "Newly registered domain behaviour analysis." (Date accessed: 30/05/2024). (), [Online]. Available: https://gitlab.com/Fatkid/newly-registered-domain-behaviour-analysis.

Appendix A

Source Code

Listing A.1: Downloads the NRDs from WhoisDB

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
import os
import time


path = os.path.abspath(os.path.join(os.path.dirname( __file__ ), '..', 'new_reg_dom'))
options = Options()
options.add_argument("—headless")
options.add_argument("start-maximized")
#accept cookies
options.add_experimental_option("prefs", {"profile.default_content_setting_values.cookies": 2,

                                    'download.default_directory': "/home/user/Desktop/DNS/newly_reg_domains"})
# installing the instance of browser
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)
# browser.implicitly_wait(5)


try:
    #open website with browser
    driver.get('https://www.whoisds.com/newly-registered-domains')


    #To see all components of website you press CTRL+U and then you find the button or link that you want to hit

    download_buttons = WebDriverWait(driver, 10).until(EC.presence_of_all_elements_located((By.XPATH,
    '//button[@class="btn btn-primary btn-xs"]')))


    if download_buttons:

        download_buttons[0].click()

        WebDriverWait(driver, 30).until(lambda x: x.find_element(By.XPATH,
        '//button[@class="btn btn-primary btn-xs"]'))


except Exception as e:
    print(str(e))


time.sleep(15)


driver.quit()
```

Listing A.2: Collects the Resource Records from NRDs

```
import json
import dns.resolver
import time
from datetime import date
from datetime import timedelta

today = date.today()
yesterday = today - timedelta(days = 1)

def query_dns_records(domain_names):
  """Queries DNS records for a list of domain names.

  Args:
    domain_names: A list of domain names.

  Returns:
    A list of DNS records, one for each domain name in the input list.
  """

  records = {}
  for domain_name in domain_names:
    resolver = dns.resolver.Resolver()
    resolver.timeout = 20
    resolver.lifetime = 80
```

```
    try:
      a_records = resolver.resolve(domain_name, "A")
      mx_records = resolver.resolve(domain_name, "MX")
      ns_records = resolver.resolve(domain_name, "NS")
      txt_records = resolver.resolve(domain_name, "TXT")
      soa_records = resolver.resolve(domain_name, "SOA")
      aaaa_records = resolver.resolve(ns_records[0].to_text(), "AAAA")

      record = {
        domain_name:{
          "records":[
            {"date": str(today),
              "A": [a_record.to_text() for a_record in a_records],
              "AAAA": [aaaa_record.to_text() for aaaa_record in aaaa_records],
              "MX": [mx_record.to_text() for mx_record in mx_records],
              "NS": [ns_record.to_text() for ns_record in ns_records],
              "TXT": [txt_record.to_text() for txt_record in txt_records],
              "SOA": [soa_records[0].to_text()],
            }
          ]
        }
      }

      records.update(record)
    except (dns.resolver.NoAnswer, dns.resolver.NXDOMAIN, dns.resolver.NoNameservers, dns.resolver.LifetimeTimeout):
      continue

  return records

def main():
  # Read the domain names from a file.
  with open("/home/user/Desktop/DNS/Random_domains/random_domains_"+ str(yesterday) + ".txt" , "r") as f:
    domain_names = f.read().splitlines()

  # Query the DNS records for the domain names.
  records = query_dns_records(domain_names)

  #Produce output of the records in the same table in a JSON format.
  with open("/home/user/Desktop/DNS/json_records_domains/records_"+str(today)+".json", "a") as outfile:
        json.dump(records, outfile, indent=4)

if __name__ == "__main__":
  main()
```

Listing A.3: Scans IPs through AbuseIPDB, Greynoise and FireHOL Databases

```
import shodan
from greynoise import GreyNoise
import requests
import json
import ipaddress
from datetime import date, timedelta
import os

def check_ip_in_firehol(ip):
    firehol_lists_directory = "/home/user/Desktop/DNS/firehol_blocklists/blocklist-ipsets-master"
    for filename in os.listdir(firehol_lists_directory):
        if filename.endswith(".ipset"):
            with open(os.path.join(firehol_lists_directory, filename), "r") as file:
                if ip in file.read():
                    return True
    return False

today = date.today()
yesterday = today - timedelta(days = 1)

SHODAN_API_KEY = "API_KEY"
GREY_API_KEY = "API_KEY"
ABUSEIP_API_KEY = "API_KEY"

with open("/home/user/Desktop/DNS/IPs/ips"+str(yesterday)+".txt", "r") as file:
    ips = file.read().splitlines()

count = 0 #counter for checked IPs

def shodan_search(target):
    s_api = shodan.Shodan(SHODAN_API_KEY)
    try:
        host = s_api.host(target, history=True)
        with open("/home/user/Desktop/DNS/scan_ips/shodan_data_"+str(yesterday)+".json", "a") as outfile:
            json.dump(host, outfile, indent=4)
    except shodan.APIError as error:
        if error == "No information available for that IP.":
            pass
        else:
            pass


def grey_search(target):
    grey_api = GreyNoise(api_key=GREY_API_KEY, offering='community')
    try:
        t = target.strip('\\n')
        res = grey_api.ip(t)
        with open("/home/user/Desktop/DNS/scan_ips/greynoise_data_"+str(yesterday)+".json", "a") as outfile:
            json.dump(res, outfile, indent=4)
```

```
    except Exception as error:
        pass

def abuseip_search(target):
    url = 'https://api.abuseipdb.com/api/v2/check'
    t = target.strip('\\n')
    if ipaddress.ip_address(t).is_private is False:
        headers = {
            'Key': ABUSEIP_API_KEY,
            'Accept': 'application/json',
        }
        params = {
            'maxAgeInDays': 200,
            'ipAddress': t,
            'verbose': ''
        }
        r = requests.get(url, headers=headers, params=params)
        json_Data = json.loads(r.content)
        if 'errors' in json_Data:
            print(f"Error: {json_Data['errors'][0]['detail']}")
            exit(1)
        else:
            with open("/home/user/Desktop/DNS/scan_ips/abuseIP_data_"+str(yesterday)+".json", "a") as outfile:
                json.dump(json_Data, outfile, indent=4)

def save_firehol_results(ips):
    with open("/home/user/Desktop/DNS/scan_ips/firehol_results_"+str(yesterday)+".txt", "a") as output_file:
        for ip in ips:
            if check_ip_in_firehol(ip):
                output_file.write(ip + "\n")

for ip in ips:
    count += 1
    shodan_search(ip)
    grey_search(ip)
    abuseip_search(ip)

save_firehol_results(ips)

print("Number of checked IPs: "+ str(count))
```

Listing A.4: Scans Domains through the Spamhaus API

```
from datetime import datetime
import requests
import time
from datetime import date
from datetime import timedelta

today = date.today()
yesterday = today - timedelta(days = 1)

# get the start time
st = time.time()

print(datetime.now())

def spamhaus_black():
    mal_num = 0
    non_mal_num = 0
    result_string = ""

    for i in blcklist[9:]:
        j = i.strip('127.0.0.1   ')
        req = requests.get('https://apibl.spamhaus.net/lookup/v1/dbl/' + j, headers=headers)

        if req.status_code == 200:
            mal_num += 1
            result_string += f"I found {mal_num} malicious domains\n"
            jsonResponse = req.json()
            x = jsonResponse["resp"][0]
            if x in resp_200_dict.keys():
                result_string += f"{j} {resp_200_dict.get(jsonResponse['resp'][0])}" + "\n"
                result_string += req.content.decode('ascii') + "\n"
        elif req.status_code != 404:
            result_string += f"{j}  {req.status_code} {resp_dict[req.status_code]}" + "\n"
        else:
            non_mal_num += 1

    result_string += f"I found {mal_num} malicious websites and {non_mal_num} non malicious or not listed"
    return result_string


def spamhaus_nrd():
    mal_num = 0
    non_mal_num = 0
    malist =[]
    for nrd in nrdlist:
        req = requests.get('https://apibl.spamhaus.net/lookup/v1/dbl/' + nrd, headers=headers)
        if req.status_code == 200:
            mal_num += 1
            print(f"I found {mal_num} malicious domains")
            jsonResponse = req.json()
            x = jsonResponse["resp"][0]
            if x in resp_200_dict.keys():
```

```
                malist.append(nrd + " " + resp_200_dict.get(jsonResponse["resp"][0]))
                print(req.content.decode('ascii'))
        elif req.status_code != 404:
            print(f"{nrd}  {req.status_code} {resp_dict[req.status_code]}")
        else:
            non_mal_num += 1
            print(f"{nrd}  {req.status_code} {resp_dict[req.status_code]}")
    return f"Today {datetime.now()} I found {mal_num} malicious websites which are {malist} and {non_mal_num}
    non malicious or not listed"

with open("/home/user/Desktop/DNS/Random_domains/random_domains_"+str(yesterday)+".txt", "r") as f:
    nrdlist = f.read().splitlines()
with open("/home/user/Desktop/DNS/Random_domains/random_domains_"+str(yesterday)+".txt", "r") as f:
    blcklist = f.read().splitlines()

# create a dictionary for the request responses of Spamhaus
resp_dict = dict({200: "OK - At least one record was FOUND",
                  400: "Bad request - there was a syntax error in the request",
                  401: "Authorization failed - please verify a valid DQS key was supplied",
                  403: "Forbidden - Authorization denied",
                  404: "Not found - The record is not listed",
                  406: "Not Acceptable - The requested Content-Type is not supported.",
                  429: "Too Many Requests - Rate limiting in effect, please decrease query rate",
                  504: "Gateway timeout - Query could not be successfully sent"})

# create a dictionary with dict() for the 200 code of response
resp_200_dict = dict({2002: "Domain used for spam",
                      2003: "Spam domain used as a redirector / URL shortener",
                      2004: "Phishing domain",
                      2005: "Malware domain",
                      2006: "Botnet C & C domain",
                      2102: "Origin domain of abused - legit spam",
                      2103: "Origin domain of abused redirector / URL shorteners used for spam",
                      2104: "Abused - legit phishing domain",
                      2105: "Abused - legit malware domain",
                      2106: "Origin domain of abused - legit botnet C & C",
                      3002: "Domain listed in Spamhaus ZRD first observed between 0 and 2 hours ago.",
                      3003: "Domain listed in Spamhaus ZRD first observed between 2 and 3 hours ago.",
                      3004: "Domain listed in Spamhaus ZRD first observed between 3 and 4 hours ago.",
                      3023: "Domain listed in Spamhaus ZRD first observed between 22 and 23 hours ago.",
                      3024: "Domain listed in Spamhaus ZRD first observed between 23 and 24 hours ago."})

headers = {'Authorization': 'k4m6dhfebzqkfyvslcz62jgrz4'}
choice = input("Please choose if you want to check the blacklist or the newly registered domains by pressing b or n?\n")
if choice == "b":
    report = spamhaus_black()
    #print(report)
    with open("/home/user/Desktop/DNS/spamhaus_reports/report_"+str(yesterday)+".txt", 'a', encoding='utf-8') as f:
        f.write(report+"\n")
elif choice == "n":
    report = spamhaus_nrd()
    print(report)
    with open("/home/user/Desktop/DNS/spamhaus_reports/report_"+str(yesterday)+".txt", 'a', encoding='utf-8') as f:
        f.write(report + "\n")

# get the end time
et = time.time()

# get the execution time
elapsed_time = et - st
print('Execution time:', elapsed_time, 'seconds')
```

Listing A.5: WhoIS lookup script

```
import socket
from datetime import date
from datetime import timedelta


today = date.today()
yesterday = today - timedelta(days = 1)


def domain_lookup(dm: str):

        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(("whois.iana.org", 43))
        s.send(f"{dm}\r\n".encode())
        response = s.recv(4096).decode()
        s.close()
        return response


def main():
        with open("/home/user/Desktop/DNS/Random_domains/random_domains_"+ str(yesterday) + ".txt" , "r") as f:
                domain_names = f.readlines()

        for domain_name in domain_names:
                domain_name = domain_name.strip()
                info = domain_lookup(domain_name)

                with open("/home/user/Desktop/DNS/whois/whois_"+str(yesterday)+".txt", "a") as outfile:
                        outfile.write("Domain: "+ domain_name + "\nWHOIS info:\n" + info + "\n")
```

```
if __name__ == "__main__":
        main()
```

Listing A.6: Entropy calculation script

```
import re
import math
from collections import Counter

def extract_domains(input_file):
    domains = []
    with open(input_file, 'r') as file:
        lines = file.readlines()

    i = 0
    while i < len(lines):
        line = lines[i].strip()
        if line.startswith("Domain:"):
            domain = line.split("Domain: ")[1]
            domains.append(domain)
            i += 3  # Move to the next block (skipping Organization and Address)
        else:
            i += 1

    return domains

def entropy (domain):
    trimmed_domain = ''.join(domain.split('.')[0:-1]) #consider the domain without dots and without the TLD label
    alphabet_counter_single =  dict()
    #calculate the character frequency
    all_char=0
    for ch in trimmed_domain:
        all_char+=1
        if ch in alphabet_counter_single:
            alphabet_counter_single[ch] += 1
        else:
            alphabet_counter_single[ch] = 1

    freq_alphabet_single = {k: float(v)/all_char for k, v in alphabet_counter_single.items()}
    #calculate the entropy based on each character frequency
    entropy = 0.0
    for ch in set(trimmed_domain):
        entropy = entropy + freq_alphabet_single[ch] * math.log(freq_alphabet_single[ch], 2)
    entropy = -entropy
    return entropy

def main():
    input_file = '/home/user/Desktop/DNS/domain_names.txt'
    output_file = '/home/user/Desktop/DNS/entropy_results_table_2.txt'

    # Extract domain names from input file
    domain_names = extract_domains(input_file)

    # Calculate entropy for each domain name
    with open(output_file, 'a') as out_file:
        # Write header for the table
        out_file.write("Domain\tEntropy (bits per character)\n")

        for domain in domain_names:
            calc_entropy = entropy(domain)
            out_file.write(f'{domain}\t{calc_entropy:.2f}\n')

    print(f"Entropy calculations saved to '{output_file}'.")

if __name__ == '__main__':
    main()
```