Subdomain Leakage: Investigating Exposure Methods and Malicious Exploitation

1st Koymans, David
Master Security an Network Engineering
University of Amsterdam
Amsterdam, Netherlands
david.koymans@os3.nl

2nd Toorop, Willem
NLnet Labs
Amsterdam, Netherlands
willem@nlnetlabs.nl

3rd Hove, Koen van
NLnet Labs
Amsterdam, Netherlands
koen@nlnetlabs.nl

***Abstract*— Subdomain enumeration is an essential part of the reconnaissance phase; it involves the accumulation of as many subdomains as possible within a domain and is extensively employed to increase the attack surface of a target. Exposed subdomains might divulge sensitive information or be susceptible to subdomain takeovers. This paper investigates the mechanism of subdomain leakage, and develops a taxonomy that correlates exposure methods with discovery techniques. Four categories are derived from relevant literature: *Indexing*, *Generating*, *Exploiting*, and *Trapping*. To provide greater insight into how subdomain enumeration is applicable in the real world, this thesis adopts a novel approach by intentionally leaking subdomains through various means and monitoring the period, nature, and frequency of subsequent attacks. Based on the results obtained from these honeypots, this thesis concludes that keeping a subdomain private is virtually impossible, highlighting the need for domain administrators to be aware of the associated risks. However, exposure can be minimised by using DNS wildcards, robust security measures, and smart naming schemes.**

***Index Terms*—subdomain enumeration, DNS, honeypots, subdomain leakage**

I. Introduction

The *Domain Name System (DNS)* is an essential part of the backbone of the World Wide Web. Through DNS, a hierarchical mapping is created between a string identifier (domain name) and an IP address [14]. While DNS entries, also called *Resource Records (RRs)*, may include additional information and resources, their primary function remains translating between mnemonic domain names and IP addresses. To achieve a global scale, DNS implements a hierarchical structure enabling a distributed system in which domain names are divided into domains and subdomains. In this scenario, an organization can own a domain and create associated subdomains as required. Through delegations, we can locate a DNS server that is authoritative for a given domain. Essentially, a subdomain can represent various things. A common practice is to utilize subdomains to separate specific functionalities within a website, such as *blog.website.com*, *contact.website.com*. Moreover, subdomains occasionally contain services or information associated with an individual or entity. Additionally, subdomains often correspond to a service hosted by an organization, whether for internal or external use. For instance: *webmail.website.com*, *calendar.website.com*, or *api.website.com* [43][10].

Special thanks to NLnet Labs for accommodating this research

Subdomain enumeration is the process of collecting as many subdomains belonging to a given domain. Through this, the attack surface can be increased significantly. It might expose a neglected or rarely used service with misconfigured or weak security, which could serve as an entry point for potential attacks [16].

The aggregation of information about a system, whether for pen-testing or more nefarious reasons, is an essential part of the reconnaissance phase [40]. This concept is often referred to as footprinting [23], in this context through subdomain enumeration. Due to this being a common practice, numerous freely available tools and services exist. Tools such as *Amass [4], Subfinder [33], Knockpy [15], altdns [21]* and *sublist3r [2, 23, 25]* employ various techniques to enumerate over a domain. These tools utilise a plethora of sources and methods. While each tool is distinct in its approach, considerable overlap exists. A dichotomy can be made between two types of sources: passive and active [41].

Within passive acquisition, as the name implies, passive techniques are used to create a footprint. This evades direct interaction and heavily reduces the chance of detection by the target system. These techniques employ known databases, search engines, and other OSINT techniques [44].

Active sources, on the other hand, rely on interaction with the target system. While they might notify a target about a potential attack, they usually yield better results. Active sources are more prominent, ranging from context-based probing to brute forcing and exploitation of known DNS vulnerabilities.

A. Risks

Once a subdomain has been leaked through any of the aforementioned methods, several risks emerge. The most significant factor is the increase in the attack surface. This larger attack surface increases the likelihood of an exploitable vulnerability being discovered [23]. Additionally, they can potentially be exploited through subdomain takeover, making users vulnerable to phishing, cross-site scripting, and account takeover [35].

When considering subdomains related to entities, data leakage can also become an issue. This could either be proprietary

company information, or personal data such as customer names [36][46].

The role of subdomain enumeration in exposing these risks is more prevalent in *IPv6*. Within *IPv4*, services can additionally be discovered by scanning the entire Internet for a specific port. Which is feasible because of the limited IP space of *IPv4*. With *IPv6*, IP scanning is no longer feasible. Therefore, it is easier to discover services hosted solely on *IPv6* by using approaches such as DNS enumeration [8][22].

B. Contributions

This research identifies how subdomains are leaked and explores potential countermeasures. The experiment consists of intentionally leaking subdomains and monitoring how, when, and if they are exploited. In contrast to existing literature, this research takes a different approach by monitoring subdomain enumeration in real-world scenarios. More specifically, this research addresses the following research questions:

1) How do different subdomain leaking techniques differ regarding the timing of their discovery and exploitation?
2) What vulnerabilities should domain administrators be mindful of to prevent subdomain leakage?

C. Related Literature

Extensive literature already exists on the topic of subdomain enumeration. These papers often take the point of view of a pen-tester during the reconnaissance phase. The existing literature can be split into two groups: *identification methods* and *analysis methods*.

Research on *identification methods* focuses on new and improved techniques for discovering subdomains. Research as performed by Marchal et al. [28] employs a semantic-based approach to optimally find new subdomains by leveraging knowledge of existing subdomains. I.e., if the subdomains *London* and *Amsterdam* exist, *Berlin* is more likely to exist than *Mandarin*. Similarly, Degani et al. [11] presents a novel machine-learning-based approach to generate candidate subdomains using existing domain knowledge. Furthermore, Ramadhan, Aresta, and Hariyadi [34] developed a new tool called *Sudomy* which uses both *active* and *passive* information gathering techniques.

Alternatively, research on *analysis methods* examines existing tools and techniques on effectiveness and prevalence. Skwarek et al., Rashid, Kamrul, and Islam explore prominent exposure methods within the context of the associated risks. Kathrine, Baby, and Ebenzer [23] and Shivananjappa and Creutzburg[42] both provide an overview of state-of-the-art subdomain enumeration tools, going into depth regarding the sources each tool uses. Lastly, [9] explores the prevalence of misconfigured AXFR transfers within real-world zone files.

Both groups provide extensive insight into subdomain enumeration and exposure. However, they fall short in showing what is being deployed in a real environment. While Marchal et al. and Degani et al. show significant results, their approaches are extremely resource-intensive. Within *identification methods*, this is a common limitation, and these papers lack insight into whether these new methods are actually utilised by malicious parties, and thus pose a real risk. Meanwhile, *analysis methods* focus more on related risks, often highlighting real-world examples of misconfigurations and the effectiveness of existing tools. However, they also lack concrete data regarding the prevalence of these methods from the perspective of a domain administrator.

This literature review reveals a significant gap in the literature: studies connecting the theoretical methods of enumeration with real-world observations. This study aims to bridge this gap by utilising honeypots to gather numerical real-world data, which can be connected to enumeration techniques. This is a similar approach as the one taken by Goseva-Popstojanova et al. They focus on how web services are discovered and consequently attacked through port discovery methods. This research, however, does not rely on port discovery but on discovery through DNS subdomain enumeration, which is an approach that, according to current findings, has not yet been pursued. Through this, a contribution is made not only by highlighting the various methods for subdomain enumeration, but also by exploring what methods are employed by malicious actors. This will provide a better perspective on how these entities operate, enabling more direct and concrete countermeasures.

II. Taxonomy of Leaks

In addition to grouping sources into active and passive acquisition techniques, this paper proposes a taxonomy to categorise sources into four groups. This grouping is based on how the method of exposure correlates with the discovery. The four categories are: *Indexing, Exploiting, Trapping* and *Generating*. During this paper, all discussed sources will be analysed in correlation to their category.

A. Indexing

As the name implies, within this category, subdomains are leaked through methods that make them indexable. The most prominent example of this is a private subdomain that is indexed in the Google search engine. There are several potential ways this can occur, most often without an individual being aware of the consequences. For example, unrestricted crawlers might map more of the infrastructure through hyperlinks than intended. Additionally, websites such as *SSLLabs*[1] will cause a URL to be indexed if sharing is enabled. Other factors, such as having a link to a subdomain on GitHub or posting it to a public forum, are also prevalent.

The procedures to find these subdomains are all based on passive acquisition techniques. An example of one method is *Google Dorks* as shown by Abasi, Farooq, and Hakkala[1]. Using this method, it becomes trivial to find and extract all associated subdomains of a domain when indexed. Thus, indexing is a serious risk and a common factor in undesired subdomain exposure.

The responsibility of preventing these leaks is shared between the system administrator and the subdomain's users. If

[1]https://www.ssllabs.com/ssltest/

a link to a subdomain is shared online, it can be discovered. This sharing can be caused by both users and administrators. However, the predominant responsibility lies with the administrator, since indexing typically pertains to management rather than usage.

B. Exploiting

Exploiting relates to sources that abuse systems and DNS functionalities to reveal more information than intended. While they are not necessary exploits in the literal sense that they abuse bugs, instead, they relate to information that is exposed through misconfiguration. A good example of this is zone transfers, referred to as *AXFR*. Zone transfers are an essential part of the DNS infrastructure, allowing servers to copy the zone file of another server, enabling scalability and resilience. If a server is configured to allow unrestricted access to zone transfers, all contained records can be leaked. [9] demonstrates the effectiveness and prevalence of subdomain discovery using this technique.

Similar to this is reverse DNS. It facilitates a reverse lookup from an IP address to the associated domain. While it is good practice to translate each IP address to a domain, there are risks associated with exposing private subdomains through it.

DNS has many features attempting to prevent information disclosure. For instance, wildcards complicate subdomain enumeration, as every query now returns a valid response. However, these defensive methods usually require explicit configuration. To prevent malicious entities from replaying responses to deny a service [12]. DNSSEC introduced the concept of *denial of existence*, which enables DNS resolvers to confirm that the requested subdomain provably does not exist [37][39][38]. It does this by answering with the alphabetically ordered existing previous subdomain and next after subdomain, thus validating that nothing exists in between. This functionality is called *NSEC*, and is an essential part of DNS. However, since every non-existent DNS request made by a resolver is answered with existing subdomains, this facilitates domain-walking, making it trivial to discover all subdomains within a given domain. To mitigate this, NSEC3 was introduced, which utilizes cryptographic hashes of the previous and next existing subdomains [6].

To exploit any of these misconfigurations, active enumeration is required. All methods of active enumeration depend on some form of interaction with the target system, although the extent varies heavily. While *AXFR* zone transfers result in explicit log entries, domain walking through *NSEC* is much more discreet

The only passive technique in this category is the utilisation of SSL Certificates. If a service is hosted on a subdomain that necessitates TLS-based communication, a certificate is required and must be requested. As of 2012, every requested certificate is documented through *certificate transparency (CT)*. This is an initiative, originally proposed by Google, to log all requested SSL Certificates in a publicly accessible database [26]. When each subdomain has its own certificate instead of wildcard certificates, this could result in the exposure of information through passive querying, as shown by Roberts and Levin[36].

All the above concepts rely on the system administrator to properly configure and secure the related infrastructure. Therefore, the prevention of subdomain leakage within this category is fully the responsibility of system administrators.

C. Trapping

Trapping is the process by which subdomains are collected through malicious services. These services present themselves as trustworthy and operate correctly, while collecting data without an individual's knowledge. This category additionally encompasses parties that offer a genuine service, but collect and share more data than the user is aware of. Examples are: *VPNs* [24], *Open resolvers*, *ISPs* [31], etc.

While both users and system administrators may fall victim to this, the leaks depend on active interaction with the subdomain. Thus, users are the more likely victims.

Since the trapping entities collect the data shared with them and do not directly interact with the affected system, this method only contains passive techniques.

D. Generating

The final category is generating. This is the process of guessing a logical subdomain. While these subdomains are never explicitly leaked, they can still be exposed through semantic analysis or simple brute-force methods. Generating often relies on conventions, such as multiple entities choosing similar names for similar services. Semantic-based generation is more complex and requires existing knowledge of the domain [11][28]. By taking known subdomains as input, predictions are made on other subdomains. After a new candidate subdomain has been generated, validating it depends on interaction with the target system. Thus, these methods of finding subdomains are active techniques and may notify the target.

III. Methodology

This section will first cover the setup created for the experiment. An extensive description will be provided regarding the infrastructure and certain design choices.

Secondly, the paper will discuss how subdomains were leaked, followed by how the accumulated data was analysed.

Lastly, the ethical implications are described, and how they were mitigated.

A. Setup

For the experiment, three domains were established: *Boulderbears.nl*, *cryptomanic.nl*, and *healthhaven.nl*. These were chosen to represent various types of organizations. *Boulderbears* is a domain focused on a hobby. These types of domains are often less professional. *Cryptomanic* was based on a financial entity, offering a crypto-exchange platform. Meanwhile, *Healthhaven* is characterized as a healthcare organization that potentially contains personal records. These three distinct entities are selected to facilitate a comparison related to a specific variant of service.

Each domain is hosted on a separate NGINX web server. These servers additionally host all associated subdomains through virtual hosts. The website design is created using generative AI with a service called *Pageflow* [32]. Each subdomain consists of a simple white page with the associated subdomain name displayed in the middle. The wildcard subdomains were treated as normal subdomains and contained a single page indicative of the wildcard.

To prevent semantic-based generation from skewing the results, as shown by Degani et al.[11], the subdomain names consist of 8-character randomly generated strings, unless the method of leaking requires a semantically relevant name. Each domain hosted the same subdomains to ensure consistent results. When a leak method required its own zone file (e.g., *unrestricted AXFR, NSEC*), the associated domains were nested one zone deeper. They are hosted using the additional subdomains: *Alpha, Bravo, Charlie, Delta* and *Echo*, based on the military alphabet. Each DNS entry receives both an A (IPv4) record and an AAAA (IPv6) record.

The DNS zones associated with each domain were all provided by a single *authoritative naming server*. The software used to provide this service was *NSD* [30], which was combined with *DNStap* [13] to log all DNS queries. The NSD server was replicated to provide redundancy. This service was hosted out of the bounds of the domains and was assigned its own subdomains: *ns1* and *ns2*. All zones were signed with NSEC3 using DNSSEC unless otherwise required.

Additionally, each domain has its own unique IPv4 and IPv6 reverse DNS zone. These zones are also signed using *DNSSEC*; however, due to a misconfiguration further up the chain of trust, these domains can not be validated with DNSSEC.

Figure 1, shows the full infrastructure. Each above-described service was hosted on a *virtual machine (VM)* with the same host. Each VM got an associated publicly routable *IPv4* and *IPv6* address. The entire infrastructure is defined within Ansible for simple deployment and repeatability. Furthermore, using Ansible ensures consistency and prevents inaccuracies in the setup from skewing the results. The full Ansible playbook is made available on request.

Since both the web servers and DNS service are self-hosted, the experiment contains two sources from which data is collected. It defines a subdomain as being *found* when, after it was exposed, a DNS query is registered within the log files of the authoritative naming server. A subdomain is *visited* if a log entry exists within the NGINX server.

All domains, along with their subdomains that host their own zone files, were made indexable by both the Google and Bing search engines.

B. Leakage Methods

Within the experiment, 27 different leaking methods were utilised. Each method consisted of three subdomains per domain, for a total of 81 subdomains per domain. By employing multiple subdomains for each leaking method, the results can be validated both within a single domain and among all

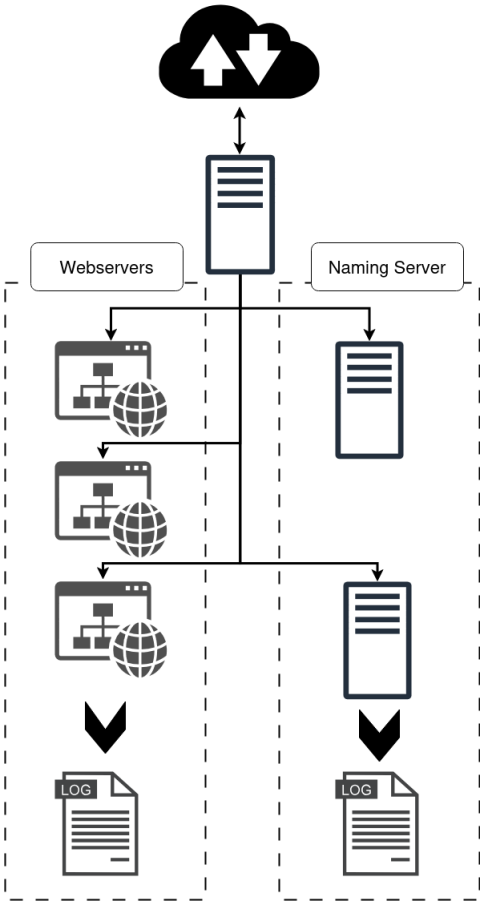

Fig. 1. Experiment Setup. The left side displays three nginx-servers, each associated with a domain and hosting all subdomains as virtual hosts. The right side displays two authoritative naming servers providing the zone files for the domains. The naming servers are replicated for redundancy. The entire setup is hosted on virtual machines, all within the same host.

domains. While certain subdomains were leaked the moment the experiment came online, some required additional steps to be taken. The time when a leak is deemed complete is recorded, and based on this, the difference between the leak occurring and its discovery is calculated. Table I illustrates all examined exposure methods, categorized depending on the defined taxonomy in section II.

Data was collected from the correlating log files over 10 days. As described in the setup, a dichotomy can be made between found and visited data. These files were parsed using Python scripts. The scripts gather the time a subdomain was first found or visited, and based on this, calculate the time before a subdomain is found.

Since certain methods of exposure also require interaction with the honeypots, the data requires sanitisation. This requires certain log file entries to be excluded from the dataset. To distinguish between these and malicious visits, entries were analysed based on access time. Additionally, IP addresses were inspected using *reverse lookup* to find associated identification using *Whois* [47]. Finally, IP addresses were also assessed on previous malicious activities using *AbuseIPDB* [3]. Based on

TABLE I
METHODS USED TO LEAK THE SUBDOMAINS

Indexing	
Sitemap	Links to subdomains within sitemap [17]
Redirect	Hyperlinks in index page
Robots.txt	Links to subdomains within robots.txt [27]
GitHub link	Links to subdomains uploaded to GitHub mimicking API endpoints [45]
SSLLabs	subdomains tested using SSLLabs
DNSTester	subdomains tested using DNSTester[a]
Exploiting	
Zone Transfer	Zone allowed AXFR from all IPs
NSEC	Zone implemented NSEC
No Wildcard	Zone did not contain wildcard
No DNSSEC	Zone was not signed with DNSSEC
Reverse IPv4	Subdomains were present in Reverse Zone (IPV4)
Reverse IPv6	Subdomains were present in Reverse Zone (IPv6)
Cert. Trans.	SSL Certificates were explicitly requested for subdomains
TextRR	DNS TXT resource record containing subdomains
Generating	
Common Service	Subdomains named after common services: *webmail*, *blog*, and *support*
Common protocol	Subdomains named after common protocols: *ssh*, *ftp*, and *telnet*
Common vuln	Subdomains named after common vulnerabilities: *admin*, *dev*, and *test*
Locations	Subdomains named after locations: *Amsterdam*, *London*, and *Berlin*
Companies	Subdomains named after companies: *asml*, *shell*, and *klm*
Country code	Subdomains named after country codes: *nl*, *en*, and *de*
Trapping	
Bad resolver	public resolver from *yandex.ru*
Online resolver	online DNS lookup from *nslookup*[b]
Free VPN	subdomains visited using *Lucky VPN UnRegister*
Visited	accessed subdomain
Control	
Control	Not leaked
NSEC3	Zone implemented NSEC3

[a]https://dnschecker.org/
[b]https://www.nslookup.io/

this, a manual consensus was reached regarding the exclusion of the data.

Furthermore, since non-malicious entities might scan the Internet for subdomains for a plethora of reasons. A distinction needs to be made regarding the intention of the visit. When a subdomain is visited, it poses a significant challenge to quantify it as malicious. The act of labelling traffic remains an open research question [20]. Within this research, a simple approach is taken based on common malicious traffic patterns. The most prevalent malicious HTTP traffic regards the enumeration of known file paths [18]. For this reason, if a subdomain at some point received a request for a path containing the string: env, PHP, API, or git, it is marked as being visited with malicious intent.

Lastly, the DNS request logs were also analysed based on the type of requests made. The same dataset was used, and within the context of the above-described taxonomy, a comparison is made between the exposure method and *resource records* requested.

C. Ethical Consideration

Because this research creates multiple honeypots, designed to give the impression of a real website, specific care must be taken to avoid non-malicious users from interacting with them. This will be achieved by having all interactive elements of the website only operate client-side, so no information is shared with the server. Additionally, all created websites will contain a visible text box, which explains the experiment and will contain a method to contact the researchers for any questions. All requests to exclude collected data will be complied with.

IV. RESULTS

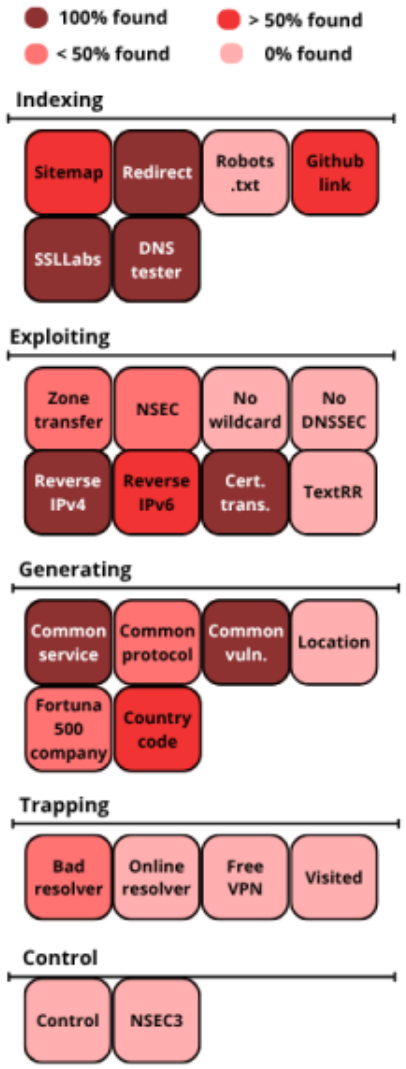

Fig. 2. The percentage of subdomains found per method, ordered in associated category

Figure 2 highlights which methods of leakage resulted in subdomains being found during the experiment, categorised within the previously defined taxonomy. A distinction is made between the number of subdomains found within a single method, grouped into: *no subdomains found*, *less than 50%*

found, *more than 50% found*, and *all subdomains found*. An additional category of *Control* is introduced; these subdomains were never leaked nor exploited, thus validating that the other subdomains were found through their intended method. Appendix A provides a comprehensive overview of each leakage method and the duration till exposure. From this figure, multiple noteworthy patterns arise. Firstly, solely because a subdomain within a method of leakage was found does not necessarily indicate that all subdomains associated with that method were discovered. A substantial portion of the leakage methods had more than one, but not all, subdomains uncovered. Although it is to be expected for methods within the *Generating* category, since it relies on the wordlist being applied, it is surprising for *NSEC*. This suggests that domains are scanned for NSEC and that these entities are aware of domain walking. However, these entities do not proceed to collect all the subdomains. This may imply that not all subdomain enumeration is harmful, it may simply be done to gather statistics about the Internet. Furthermore, it is noteworthy that while exposure occurred through GitHub. Not all subdomains, mimicking an API endpoint, were found, even though all links were included within a single file.

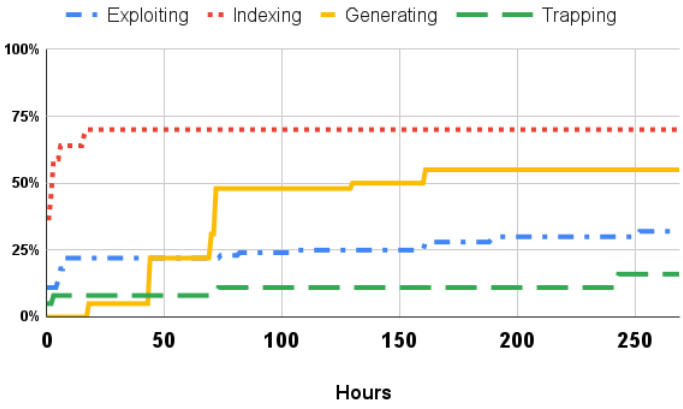

Fig. 3. Figure displays the total percentage of subdomains **found** per category over time

While Figure 2 already indicates differences among categories, Figure 3 more clearly highlights them. This figure represents the percentage of subdomains identified over a specified period, adjusted for when the subdomain was leaked. This figure indicates that by far the most prominent way subdomains are exposed is through them becoming indexable. This category likewise had its subdomains found the quickest: over 30% were found instantly after they were leaked. This demonstrates that websites such as SSLLabs and GitHub are being actively monitored for subdomains by multiple entities. Scraping is also a significant approach to find subdomains. However, it is more difficult to distinguish between scraping done by search engines or by entities with malicious intent. One method of note here is *DNStester*, this service offers DNSSEC validation services by interacting with the DNS system. While examining the domain, the service IP was

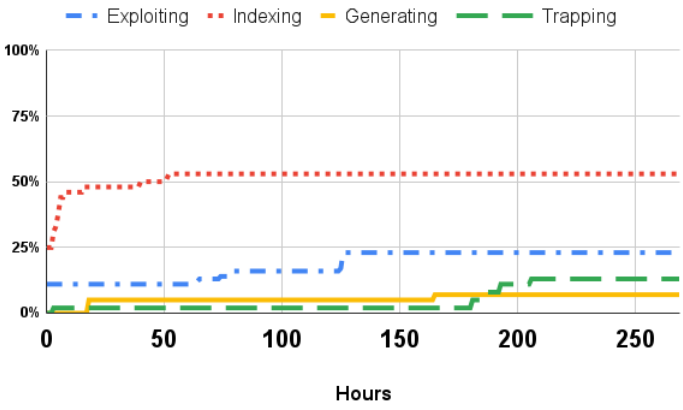

Fig. 4. Figure displays the total percentage of subdomains **visited** per category over time

properly identified. However, all nine tested subdomains would consistently be requested again after precisely three hours from different IP addresses, not identified as belonging to this service.

Compared to the *Indexing* category, the *Generate* category exhibits a more gradual exposure over time. This is presumably caused by their resource-intensive nature when compared to other categories. Subdomains named after *Common vulnerabilities*, such as *Admin, Dev* and *Test*, were found the quickest. Additionally, once a subdomain was brute-forced, most semantic based leaked subdomains were found. It is noteworthy that the domain *Boulderbears* is the only one not to have undergone brute-forcing. When combined with the stepwise increase in Figure 2, it suggests domains are not subjected to daily brute-force attacks.

A similar, although more gradual, pattern is demonstrated by the *Exploiting* category. While the majority of the found subdomains were identified within the first 24 hours, new subdomains were still being discovered after a week.

Methods categorised under *Trapping* initially exhibit minimal behaviour. However, after 200 hours, a sudden increase in activity is perceived. Speculating what caused this sudden jolt yielded no results.

Comparing Figure 3 with Figure 4, additional patterns arise. Most notable is that within every category, a distinct drop is evident between found and subsequently visited subdomains.

One explanation is that within this experiment, only visits to the ports *80* and *443* are collected since these correspond with an *NGINX* web server. Visits made based on a subdomain to another port have therefore not been gathered. The smallest drop-off exists in the *Exploiting* category.

Another oddity is the disproportional decrease within the *Generating* category. This is presumably due to the resource-intensive nature of identifying wildcards. Wildcards consistently respond to a query, even when the subdomain is non-existent, resulting in a much more resource-intensive process to distinguish existing from non-existent subdomains. Consequently, although these subdomains have been identified, it

is significantly more challenging to separate them from non-existent ones, thus resulting in fewer visits.

There is no direct correlation between a subdomain initially being found and visited. Figure 3 and 4 do not increase at equal intervals. This is largely due to the graphs only showing when a subdomain is first found or visited. When an entity identifies a subdomain without subsequently visiting, there is no information regarding when another entity finds it, nor whether the initial finder delays before visiting.

A. Webserver logs

The fact that a subdomain has been exposed and found does not directly correlate with malicious behaviour. Figure 5 displays whether visited subdomains within a category received malicious traffic. While the used condition is limited and a single malicious request is enough to categorize it as such, it provides insight into how found subdomains are potentially exploited.

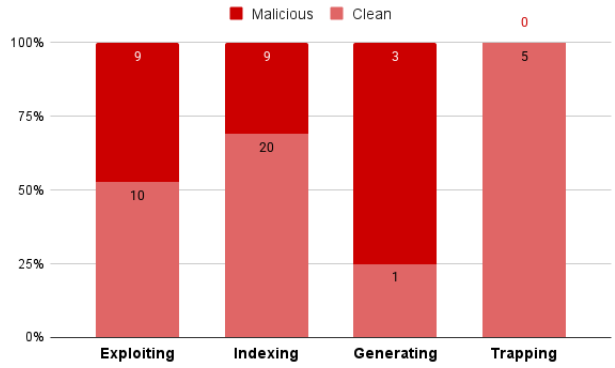

Fig. 5. Visited domains with malicious intent per category

Even though the sample size is restricted, there is evidence that certain categories are more likely to be exploited. Subdomains found through active means: *Exploiting* or *Generating* are subsequently more likely to receive malicious traffic. All the visited subdomains in the *Trapping* category were from the same method of leaking: *Bad resolver*. While it is peculiar that these subdomains are being requested 10 days later, no malicious traffic is sent against the web server.

B. DNS access logs analysis

1) Resource Records: Additional patterns can be derived when examining the requested *DNS resource records*. Figure 6 highlights the percentage of requested RRs per category. Examining *IPv4* (A) and *IPv6* (AAAA) records, the average distribution is circa 50% *IPv4* and 15% *IPv6*. Figure 6 highlights certain outliers. The *Generating* category is heavily biased towards A-records. This can potentially be explained by the fact that generating requires active probing and A records are the most prevalent, thus increasing the chances of getting a hit. Additionally, *IPv4* requires less data than *IPv6*, thereby conserving bandwidth, allowing more probes per second.

Requests made for subdomains within the *Indexing* category are more equally distributed among *IPv4* and *IPv6*.

Fig. 6. Requested resource records per category

2) Anomalies: The results encompass not only requests made for existing domains but also contain every single request directed to the DNS server, whether valid or invalid. However, because of the wildcard present, each request for a subdomain would result in a valid response. Table II contains the most requested non-existent subdomains. Notably, the most common requests consist of domains the hosted DNS server is not authoritative over. As for the domain *pizzaseo.com*, the request would always be made for the *RRSIG* record of *DNSSEC*. This behaviour could be indicative of a DNS-based DDoS attack [19]. Queries made for *version.bind* is a feature of the *Bind authoritative naming server* and will respond with the server version.

When analysing the queries for subdomains within the authoritative domain only, many requests are made for the *_dmarc* subdomain. *DMARC* is used for email verification sent from a specific domain. Although the domains defined in this experiment do not manage a mailing service, these requests are still actively being sent. Another peculiar activity is the numerous requests made for the *bneq9ke* subdomain; this subdomain was requested on average 50 times per domain on all three domains. While it is similar in appearance to the subdomains used within this experiment, it falls one character short. Furthermore, it is not akin to any existing subdomain. These types of requests consisted of a set of randomly generated characters, which align with the findings of Griffioen and Doerr. They associate these variants of requests with DNS-based DoS attacks.

In total, while 159 existing subdomains were requested, 6962 non-existing (sub)domains were queried. This further emphasises the results in Figure 2, which shows that subdomain brute-forcing is prevalent and something system administrators need to be aware of.

V. Discussion

The results indicate that attempting to keep a subdomain private is virtually impossible. Domains are consistently being subjected to both active and passive subdomain enumeration techniques. While subdomains exposed to passive sources are found quicker, active probing is also prevalent. The results

TABLE II
TOP REQUESTED NON-EXISTING (SUB)DOMAINS

All	
Subdomain	**Amount**
pizzaseo.com	5448
version.bind	502
isc.org	459
ripe.net	457
cloudflare.com	455
Subdomain only	
city.delta.cryptomanic.nl	392
_dmarc.healthhaven.nl	239
_dmarc.cryptomanic.nl	197
news.cryptomanic.nl	158
genesis.cryptomanic.nl	134

show that all existing domains are subjected to subdomain brute-forcing, irrelevant of the type and size of the domain. This reinforces the findings by Moore et al.[29]. Furthermore, this paper suggests that Internet background noise [7] not only relates to path enumeration [18] but also to subdomain enumeration. This principle can be described as the search for low-hanging fruit, where malicious parties pursue easy targets. Because of the arbitrary period it takes to get a domain indexed, which heavily influences the results, no results are significant regarding the difference among the domains. However, it is expected that chance plays a bigger factor than the type of service hosted within these large-scale attacks.

Based on the DNS access log, the presence of semantically similar yet non-existent subdomains suggests semantic-based generation being utilised [28]. However, it is also possible these subdomains are semantically similar by coincidence, since random sets of characters are frequently employed in DoS attacks [19].

Section IV-B1 reveals that once a subdomain has been indexed, a nearly equal distribution of requests is made for IPv4 and IPv6. Since these subdomains are indexed and thus prone to being crawled, this phenomenon could indicate crawlers prefer using IPv6 over IPv4 to avoid getting banned.

As mentioned in section IV, only ports *80* and *443* were monitored for traffic. Any traffic directed to subdomains on ports other than these is excluded from the results. While this causes an absence of numerical data, it is possible to speculate, since many DNS queries were made for mail-related services, such as requests for *_dmarc*, traffic to mail-related ports is expected.

Comparing Figure 3 and 4 in Section IV, the effect of using DNS wildcards becomes prevalent. While the subdomains have technically been found, since a DNS request was made for the associated name, the resource-intensive nature of distinguishing wildcard responses from actual responses results in almost no visits. This indicates that large-scale subdomain enumeration does not allocate resources to wildcard verification. Random names could limit the reach of the wordlist enumeration attacks. However, DNS was invented to enhance access to services via mnemonic domain names; thus, choosing complex names is counterproductive.

While the runtime of the experiment was limited, it already demonstrates how quickly subdomains are identified and consequently exploited. Since subdomains continue to be discovered 10 days into the experiment, it is reasonable to assume that a longer exposure period will result in additional subdomains being found. The limited exposure through *zone transfers* and *NSEC* is surprising, since many publicly available tools exist that actively exploit them [23]. This suggests these tools are not being actively utilized for large-scale subdomain enumeration. Thus, a more targeted reconnaissance would yield even more results. Additionally, in this setup, each method of leaking had its own associated domain. Within a normal environment, a single subdomain can fall victim to any category of exposure.

The unintended exposure of subdomains remains a significant transgression, contributing to many data leaks and security breaches [16]. It has additionally led to privacy-sensitive information being divulged [36][5] and, in some cases, to subdomain takeover, which can have severe consequences [35]. This highlights the prevalent ignorance among domain administrators about the risks of private subdomains and the false sense of security they may provide. This research emphasizes that keeping subdomains private is nearly unattainable and should not be relied on as a security principle. Domain administrators must adopt more robust security measures and develop a deeper understanding of the vulnerabilities and associated risks.

Based on these findings, the following recommendations for DNS administrators are defined:

- Use wildcard records
- Securely configure DNS services
- Expect subdomain exposure, and be aware of the associated risks
- Where possible, only provide services through *IPv6*
- Utilise smart naming schemes
- Do not insert sensitive data in subdomain names

VI. CONCLUSION

To conclude, this research examined how different subdomain leaking methods vary in terms of the timing of their discovery and subsequent exploitation. While not every method of exposure was found within the timeframe of this research, it is evident that retaining a subdomain as private is virtually impossible and should not be used as a security measure. Domain administrators should recognize that assuming malicious entities are not aware of the existence of a subdomain poses a severe security risk.

To mitigate the impact of subdomain leakage, proactive measures should be taken through regular audits, disabling unused services, and ensuring that subdomain names do not encompass sensitive information. Thus, a solid understanding of the risks involved is necessary.

Effective measures to limit exposure include maintaining proper domain hygiene, utilizing DNS wildcards, implementing appropriate security measures, and adopting a smart naming scheme. Whenever feasible, *IPv6* could be used, as this

appears to be the most effective way to keep subdomains private.

A. Future work

Additional research is required to further bridge the gap between the theoretical exploitation and real-world observation. Future work could examine more potential sources through which subdomains are leaked. *Packet capturing*, *Server Name Identification (SNI)*, *Webarchive*, etc. [34][42] could be examined on how proliferate they are. Additionally, new insight can be gained from running the experiment for a longer period and examining how exposure evolves. This would allow for differences among the types of domains to identify themselves. Finally, future research could explore to what extent subdomains are enumerated within a more pertinent domain

References

[1] Reza Abasi, Ali Farooq, and Antti Hakkala. "Google dorks: Use cases and Adaption study". PhD thesis. Master's thesis, University of Turku, 2020.

[2] aboul3la. *sublist3r*. https://github.com/aboul3la/Sublist3r. 2020.

[3] AbuseIPDB. *AbuseIPDB*. 2025. URL: https://www.abuseipdb.com/ (visited on 07/02/2025).

[4] OWASP Amass. *Amass*. Version 4.2.0. URL: https://owasp.org/www-project-amass/.

[5] Anonymousshetty. *How I Discovered a Private Key Leak on a Nokia's Subdomain*. Tech. rep. Medium, 2024. URL: https://medium.com/@anonymousshetty2003/how-i-discovered-a-private-key-leak-on-a-companys-subdomain-929100e7a561.

[6] Roy Arends et al. *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*. RFC 5155. Mar. 2008. DOI: 10.17487/RFC5155. URL: https://www.rfc-editor.org/info/rfc5155.

[7] Xavier Bellekens. *How Internet Background Noise Makes Cybersecurity Hard?* Tech. rep. Lupovis, 2022. URL: https://www.lupovis.io/how-internet-background-noise-makes-cybersecurity-hard/ (visited on 02/04/2025).

[8] Clinton Carpene. "Using passive and active enumeration methods to improve IPv6 host enumeration search algorithms". In: (2015).

[9] "Characterizing Vulnerability of DNS AXFR Transfers with Global-Scale Scanning". eng. In: *2019 IEEE Security and Privacy Workshops (SPW)*.

[10] danielmiessler. *SecLists/DNS*. https://github.com/danielmiessler/SecLists/tree/master/Discovery/DNS. 2025.

[11] Luca Degani et al. "Generative adversarial networks for subdomain enumeration". In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. SAC '22. Virtual Event: Association for Computing Machinery, 2022, pp. 1636–1645. ISBN: 9781450387132. DOI: 10.1145/3477314.3506967. URL: https://doi.org/10.1145/3477314.3506967.

[12] the DNS institute. *Proof of Non-Existence (NSEC and NSEC3)*. URL: https://dnsinstitute.com/documentation/dnssec-guide/ch06s02.html.

[13] DNStap. *golang-dnstap*. Version 0.4.0. Feb. 9, 2021. URL: https://dnstap.info/.

[14] *Domain names - implementation and specification*. RFC 1035. Nov. 1987. DOI: 10.17487/RFC1035. URL: https://www.rfc-editor.org/info/rfc1035.

[15] foweb. *knock*. https://github.com/guelfoweb/knock. 2024.

[16] Daniel Golightly. *PSA: T-Mobile subdomain leaked user account details in april*. Tech. rep. Android Headlines, 2018.

[17] Google. *What is a Sitemap*. Tech. rep. Google, 2025. URL: https://developers.google.com/search/docs/crawling-indexing/sitemaps/overview (visited on 07/03/2025).

[18] Katerina Goseva-Popstojanova et al. "Characterization and classification of malicious Web traffic". eng. In: *Computers & security* 42 (2014), pp. 92–115. ISSN: 0167-4048.

[19] Harm Griffioen and Christian Doerr. "Taxonomy and adversarial strategies of random subdomain attacks". In: *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE. 2019, pp. 1–5.

[20] Jorge Luis Guerra, Carlos Catania, and Eduardo Veas. "Datasets are not enough: Challenges in labeling network traffic". eng. In: *Computers security* 120 (2022), pp. 102810–. ISSN: 0167-4048.

[21] infosec-eu. *altdns*. https://github.com/infosec-au/altdns. 2025.

[22] Abhinav Kamra et al. "The effect of DNS delays on worm propagation in an IPv6 Internet". In: *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* Vol. 4. IEEE. 2005, pp. 2405–2414.

[23] G Jaspher Kathrine, Ronnie T Baby, and V Ebenzer. "Comparative analysis of subdomain enumeration tools and static code analysis". In: *ISSN (Online)* (2020), pp. 2454–7190.

[24] Mohammad Taha Khan et al. "An empirical analysis of the commercial vpn ecosystem". In: *Proceedings of the Internet Measurement Conference 2018*. 2018, pp. 443–456.

[25] Ms Rini Kurian and Arunima Santhosh. "Identifying Subdomains of the Website Using SUBLIST3R and Comparing SUBLIST3R AMASS, KNOCKPY". In:

National Conference on Emerging Computer Applications. Vol. 3. 1. 2021.

[26] Ben Laurie. "Certificate transparency". In: *Communications of the ACM* 57.10 (2014), pp. 40–46.

[27] H. Zeller M. Koster G. Illyes. *Robots Exclusion Protocol*. RFC. RFC Editor, Sept. 2022. URL: https://www.rfc-editor.org/rfc/rfc9309 (visited on 02/05/2025).

[28] Samuel Marchal et al. "Semantic exploration of DNS". In: *NETWORKING 2012: 11th International IFIP TC 6 Networking Conference, Prague, Czech Republic, May 21-25, 2012, Proceedings, Part I 11*. Springer. 2012, pp. 370–384.

[29] David Moore et al. "Inferring internet denial-of-service activity". In: *ACM Transactions on Computer Systems (TOCS)* 24.2 (2006), pp. 115–139.

[30] NLnetlabs. *NSD*. Version NSD 4.12.0. Apr. 17, 2025. URL: https://www.nlnetlabs.nl/projects/nsd/about/.

[31] Paul Ohm. "The rise and fall of invasive ISP surveillance". In: *U. Ill. L. Rev.* (2009), p. 1417.

[32] pageflow. URL: https://www.pageflow.ai/#newsletter (visited on 06/28/2025).

[33] projectdiscovery. *subfinder*. https://github.com/projectdiscovery/subfinder. 2025.

[34] Rizdqi Akbar Ramadhan, Redho Maland Aresta, and Dedy Hariyadi. "Sudomy: Information Gathering Tools for Subdomain Enumeration and Analysis". In: *IOP Conference Series: Materials Science and Engineering* 771.1 (Mar. 2020), p. 012019. DOI: 10.1088/1757-899X/771/1/012019. URL: https://dx.doi.org/10.1088/1757-899X/771/1/012019.

[35] S. M. Zia Ur Rashid, MD. Imtiaz Kamrul, and Asraful Islam. "Understanding the Security Threats of Esoteric Subdomain Takeover and Prevention Scheme". In: *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*. 2019, pp. 1–4. DOI: 10.1109/ECACE.2019.8679122.

[36] Richard Roberts and Dave Levin. "When Certificate Transparency Is Too Transparent: Analyzing Information Leakage in HTTPS Domain Names". In: *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*. WPES'19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 87–92. ISBN: 9781450368308. DOI: 10.1145/3338498.3358655. URL: https://doi.org/10.1145/3338498.3358655.

[37] Scott Rose et al. *DNS Security Introduction and Requirements*. RFC 4033. Mar. 2005. DOI: 10.17487/RFC4033. URL: https://www.rfc-editor.org/info/rfc4033.

[38] Scott Rose et al. *Protocol Modifications for the DNS Security Extensions*. RFC 4035. Mar. 2005. DOI: 10.17487/RFC4035. URL: https://www.rfc-editor.org/info/rfc4035.

[39] Scott Rose et al. *Resource Records for the DNS Security Extensions*. RFC 4034. Mar. 2005. DOI: 10.17487/RFC4034. URL: https://www.rfc-editor.org/info/rfc4034.

[40] Karen Scarfone et al. "Technical guide to information security testing and assessment". In: *NIST Special Publication* 800.115 (2008), pp. 2–25.

[41] Ahmed Sheikh. "Enumeration". In: *Certified Ethical Hacker (CEH) Preparation Guide: Lesson-Based Review of Ethical Hacking and Penetration Testing*. Springer, 2021, pp. 27–34.

[42] Navaneeth Shivananjappa and Reiner Creutzburg. "Vulnerability Management Using Open-Source Tools". In: *Electronic Imaging* 36 (2024), pp. 1–8.

[43] Shweta. *What is A subdomain? Everything You Need To Know*. Tech. rep. Forbes, 2024.

[44] data.europe.eu team. *What is OSINT: Open-source intelligence?* Tech. rep. Publications Office of the European Union, 2022.

[45] TheLeaker900. *Profile*. https://github.com/TheLeaker900. 2025.

[46] Genet Thorsell. *DNS Enumeration Techniques and Characterizing DNS vulnerabilities*. 2022.

[47] Whois. *Whois lookup*. 2025. URL: https://www.whois.com/whois/ (visited on 07/02/2025).

APPENDIX

TABLE III
TIME UNTIL **DISCOVERY** PER SUBDOMAIN IN *Hours*

	Method	Subdom. 1	*Boulderbears*	*Healthhaven*	*Cryptomanic*	Subdom. 2	*Boulderbears*	*Healthhaven*	*Cryptomanic*	Subdom. 3	*Boulderbears*	*Healthhaven*	*Cryptomanic*
Indexing	**Sitemap**	whs9s1sb	*6.8*	*N.A.*	*6.4*	rq55kbxk	*3.1*	*N.A.*	*6.8*	4fi53siu	*3.6*	*N.A.*	*3.6*
	Redirect	sttzmnu3	*0.1*	*16.6*	*0.2*	aq3abapr	*0.1*	*16.6*	*0.2*	e0zxpu96	*0.1*	*16.6*	*0.2*
	Robot.txt	b6u6rize	*N.A.*	*N.A.*	*N.A.*	v9fv5wvi	*N.A.*	*N.A.*	*N.A.*	dy3doh7d	*N.A.*	*N.A.*	*N.A.*
	GitHub link	610dhsbc	*N.A.*	*N.A.*	*0.6*	q6gvawkj	*N.A.*	*N.A.*	*0.6*	d9ax4d00	*0.5*	*0.5*	*0.5*
	SSLLabs	clju0kec	*0*	*0*	*0*	kw3oe1tb	*0*	*0*	*0*	pf6lzqo6	*0*	*0*	*0*
	DNS tester	vmit6d9t	*3*	*3*	*3*	pu5fgxus	*3*	*3*	*3*	1siuwd10	*3*	*3*	*3*
Exploiting	**Zone transfer**	hohl1c1x	*N.A.*	*N.A.*	*N.A.*	v1pno865	*N.A.*	*N.A.*	*161.3*	ezo0oeub	*N.A.*	*N.A.*	*N.A.*
	NSEC	e3v9hs22	*106*	*82.5*	*74.8*	zjr6n1vv	*N.A.*	*N.A.*	*N.A.*	lqiis6uk	*N.A.*	*252.6*	*161.3*
	No Wildcard	r6vfxqj3	*N.A.*	*N.A.*	*N.A.*	81dhrs26	*N.A.*	*N.A.*	*N.A.*	myfcuzon	*N.A.*	*N.A.*	*N.A.*
	No DNSSEC	rtz1kait	*N.A.*	*N.A.*	*N.A.*	5bcsc3d7	*N.A.*	*N.A.*	*N.A.*	2ju0jv6z	*N.A.*	*N.A.*	*N.A.*
	Reverse IPv4	8g344oik	*8.5*	*5.3*	*6.1*	845rw20x	*8.5*	*5.3*	*6.1*	8i2k381p	*8.5*	*5.3*	*6.1*
	Reverse IPv6	r5daw89k	*N.A.*	*192.7*	*273.1*	beo0kxzt	*N.A.*	*N.A.*	*273.0*	c5xgnpev	*N.A.*	*189.8*	*276.9*
	Cert. Trans.	i5tdrhk7	*0*	*1.3*	*0*	egcj3z35	*0*	*0.3*	*0*	ncofd2aa	*0*	*0*	*0*
	TextRR	r5y849no	*N.A.*	*N.A.*	*N.A.*	sgnhuumk	*N.A.*	*N.A.*	*N.A.*	oalh061w	*N.A.*	*N.A.*	*N.A.*
Generating	**Common Service**	webmail	*72.7*	*44.8*	*70.3*	blog	*72.8*	*44.7*	*70.3*	support	*72.6*	*44.8*	*70.3*
	Common protocol	ssh	*N.A.*	*N.A.*	*161.3*	ftp	*72.7*	*44.7*	*130.6*	telnet	*N.A.*	*N.A.*	*N.A.*
	Common vuln	admin	*72.7*	*44.7*	*18.3*	dev	*72.7*	*44.8*	*18.3*	test	*72.7*	*44.8*	*18.3*
	Location	amsterdam	*N.A.*	*N.A.*	*N.A.*	london	*N.A.*	*N.A.*	*N.A.*	berlin	*N.A.*	*N.A.*	*N.A.*
	Companies	asml	*N.A.*	*N.A.*	*N.A.*	shell	*N.A.*	*N.A.*	*161.3*	klm	*N.A.*	*N.A.*	*N.A.*
	Countrycode	nl	*N.A.*	*N.A.*	*161.3*	en	*72.5*	*44.7*	*70.2*	de	*72.7*	*44.8*	*70.2*
Trapping	**Bad resolver**	byeqs7aa	*N.A.*	*N.A.*	*N.A.*	rxkvl5va	*243.9*	*4*	*73.6*	i6przl68	*243.9*	*N.A.*	*N.A.*
	Online resolver	kejyz066	*N.A.*	*N.A.*	*N.A.*	3wgo0eqi	*N.A.*	*N.A.*	*N.A.*	g0y0nqea	*N.A.*	*N.A.*	*N.A.*
	Free VPN	z8kpfy6f	*N.A.*	*N.A.*	*N.A.*	k8crtinx	*N.A.*	*N.A.*	*N.A.*	7n5to45k	*N.A.*	*N.A.*	*N.A.*
	Visited	up6ubuw7	*N.A.*	*N.A.*	*N.A.*	p4hsoo2f	*N.A.*	*N.A.*	*N.A.*	6y4uoulp	*N.A.*	*N.A.*	*N.A.*
Control	**Control**	iwqbfgdy	*N.A.*	*N.A.*	*N.A.*	86utpebn	*N.A.*	*N.A.*	*N.A.*	w7hy7vf9	*N.A.*	*N.A.*	*N.A.*
	NSEC3	cile9p3j	*N.A.*	*N.A.*	*N.A.*	24s214to	*N.A.*	*N.A.*	*N.A.*	q46zw80a	*N.A.*	*N.A.*	*N.A.*

TABLE IV
TIME UNTIL **VISITED** PER SUBDOMAIN IN *Hours*

	Method	Subdom. 1	*Boulderbears*	*Healthhaven*	*Cryptomanic*	Subdom. 2	*Boulderbears*	*Healthhaven*	*Cryptomanic*	Subdom. 3	*Boulderbears*	*Healthhaven*	*Cryptomanic*
Indexing	**Sitemap**	whs9s1sb	*6.8*	*N.A.*	*6.4*	rq55kbxk	*3.1*	*N.A.*	*6.8*	4fi53siu	*3.6*	*N.A.*	*3.6*
	Redirect	sttzmnu3	*51.4*	*16.6*	*8.8*	aq3abapr	*5.2*	*52.6*	*4.7*	e0zxpu96	*5.6*	*40.8*	*5.4*
	Robot.txt	b6u6rize	*N.A.*	*N.A.*	*N.A.*	v9fv5wvi	*N.A.*	*N.A.*	*N.A.*	dy3doh7d	*N.A.*	*N.A.*	*N.A.*
	GitHub link	610dhsbc	*N.A.*	*N.A.*	*0.6*	q6gvawkj	*N.A.*	*N.A.*	*0.6*	d9ax4d00	*0.5*	*0.5*	*0.5*
	SSLLabs	clju0kec	*0*	*0*	*0*	kw3oe1tb	*0*	*0*	*0*	pf6lzqo6	*0*	*0*	*0*
	DNS tester	vmit6d9t	*N.A.*	*N.A.*	*N.A.*	pu5fgxus	*N.A.*	*N.A.*	*N.A.*	1siuwd10	*N.A.*	*N.A.*	*N.A.*
Exploiting	**Zone transfer**	hohl1c1x	*N.A.*	*N.A.*	*N.A.*	v1pno865	*N.A.*	*N.A.*	*N.A.*	ezo0oeub	*N.A.*	*N.A.*	*N.A.*
	NSEC	e3v9hs22	*N.A.*	*N.A.*	*74.8*	zjr6n1vv	*N.A.*	*N.A.*	*N.A.*	lqiis6uk	*N.A.*	*N.A.*	*N.A.*
	No Wildcard	r6vfxqj3	*N.A.*	*N.A.*	*N.A.*	81dhrs26	*N.A.*	*N.A.*	*N.A.*	myfcuzon	*N.A.*	*N.A.*	*N.A.*
	No DNSSEC	rtz1kait	*N.A.*	*N.A.*	*N.A.*	5bcsc3d7	*N.A.*	*N.A.*	*N.A.*	2ju0jv6z	*N.A.*	*N.A.*	*N.A.*
	Reverse IPv4	8g344oik	*126.1*	*126.1*	*78.0*	845rw20x	*126.1*	*126.1*	*62.4*	8i2k381p	*126.1*	*126.1*	*65*
	Reverse IPv6	r5daw89k	*N.A.*	*N.A.*	*N.A.*	beo0kxzt	*N.A.*	*N.A.*	*N.A.*	c5xgnpev	*N.A.*	*N.A.*	*N.A.*
	Cert. Trans.	i5tdrhk7	*0*	*0*	*0*	egcj3z35	*0*	*0*	*0*	ncofd2aa	*0*	*0*	*0*
	TextRR	r5y849no	*N.A.*	*N.A.*	*N.A.*	sgnhuumk	*N.A.*	*N.A.*	*N.A.*	oalh061w	*N.A.*	*N.A.*	*N.A.*
Generating	**Common Service**	webmail	*N.A.*	*N.A.*	*N.A.*	blog	*N.A.*	*N.A.*	*N.A.*	support	*N.A.*	*N.A.*	*N.A.*
	Common protocol	ssh	*N.A.*	*N.A.*	*166.0*	ftp	*N.A.*	*N.A.*	*N.A.*	telnet	*N.A.*	*N.A.*	*N.A.*
	Common vuln	admin	*N.A.*	*N.A.*	*18.4*	dev	*N.A.*	*N.A.*	*18.4*	test	*N.A.*	*N.A.*	*18.4*
	Location	amsterdam	*N.A.*	*N.A.*	*N.A.*	london	*N.A.*	*N.A.*	*N.A.*	berlin	*N.A.*	*N.A.*	*N.A.*
	Companies	asml	*N.A.*	*N.A.*	*N.A.*	shell	*N.A.*	*N.A.*	*N.A.*	klm	*N.A.*	*N.A.*	*N.A.*
	Countrycode	nl	*N.A.*	*N.A.*	*N.A.*	en	*N.A.*	*N.A.*	*N.A.*	de	*N.A.*	*N.A.*	*N.A.*
Trapping	**Bad resolver**	byeqs7aa	*N.A.*	*N.A.*	*N.A.*	rxkvl5va	*187.6*	*4.0*	*181.0*	i6przl68	*208.7*	*195.8*	*N.A.*
	Online resolver	kejyz066	*N.A.*	*N.A.*	*N.A.*	3wgo0eqi	*N.A.*	*N.A.*	*N.A.*	g0y0nqea	*N.A.*	*N.A.*	*N.A.*
	Free VPN	z8kpfy6f	*N.A.*	*N.A.*	*N.A.*	k8crtinx	*N.A.*	*N.A.*	*N.A.*	7n5to45k	*N.A.*	*N.A.*	*N.A.*
	Visited	up6ubuw7	*N.A.*	*N.A.*	*N.A.*	p4hsoo2f	*N.A.*	*N.A.*	*N.A.*	6y4uoulp	*N.A.*	*N.A.*	*N.A.*
Control	**Control**	iwqbfgdy	*N.A.*	*N.A.*	*N.A.*	86utpebn	*N.A.*	*N.A.*	*N.A.*	w7hy7vf9	*N.A.*	*N.A.*	*N.A.*
	NSEC3	cile9p3j	*N.A.*	*N.A.*	*N.A.*	24s214to	*N.A.*	*N.A.*	*N.A.*	q46zw80a	*N.A.*	*N.A.*	*N.A.*