# CHAPTER 1

# INTRODUCTION

The Domain Name System is indisputably one of the most important and over-looked parts of Internet. Without DNS, the Internet as we know it today, would collapse, and we would all be licking stamps to pay our bills, driving to an actual store to purchase something, reading newspaper to see what movies were showing or buying CDs to get our music. All DNS messages are transferred unencrypted making it vulnerable to an attacker who has gained access to the communicating channel thus placing the querier at risk.

There has been minimum work carried out to provide privacy between a client and a server. DNSSEC presents response integrity by using cryptographic signatures on zones permitting end users to validate if the replies are right. However, DNSSEC fails to preserve privacy of request and response. Other work includes DNSCrypt, DNSCurve, IPSECA and Confidential DNS.

Growing concerns on privacy has led to measures like the General Data Protection Regulations (GDPR) [1] being introduced across the continent of Europe. Open Certificate authorities like Let’s Encrypt [2] provide free, automated TLS certificates leading to increasing number of websites adopting HTTPS thus, preventing eavesdroppers on the communication channel between websites and users. However, the Domain Name System that translates domain names into IP addresses still uses unencrypted communication channels by default, enabling eavesdroppers to collect DNS requests and responses made by a user.

A distributed database containing mappings of domain names to data is also a protocol for Transmission Control Protocol or Internet protocol (TCP/IP) network. The DNS deals with increasing number of Internet users around the world, enabling users to use friendly, hierarchical names to find computers on a network. It translates names like www.dot.com, into Internet Protocol (IP) addresses, such as 160.153.137.14 (or extended IPv6 addresses), thus, assisting computers to communicate with each other. It acts as a medium to access Internet applications, such as the World Wide Web (www) easily.

In the past few years, public DNS resolvers like Google, Cloudflare have become very popular unlike the resolvers provided by Internet Service Providers (ISPs). These are often located in the same autonomous system while the public resolvers can be placed anywhere in the Internet adding to the increase in eavesdroppers between the user and ISP and in underlying networks.

Public DNS resolvers add privacy but is limited to their logging policies. However, these policies do not protect the queries against eavesdroppers and there is no means to test their logging policies [3]. This paper evaluates the privacy issues in the current DNS implementation and measures to elevate security and privacy around this area through utilising the new solutions available.

## Background

There has been considerable changes since 1980s when DNS was first regulated to make connections between users and the Internet by using mnemonic names such as dot.com. Multiple advancements have been made to allow a considerable portion of internet traffic to connect via HTTPS. But, there still exists concerns inherent in forwarding cleartext queries over the wire [4].

Domain Name System Security Extension (DNSSEC) was the extension of the DNS protocol to accommodate authentication in the year 2010. DNSSEC addresses only authentication while the queries are still sent and received in clear text leaving the connection between the user and responder. In order to provide authentication and confidentiality to DNS, the International Engineering Task Force (IETF) created DPRIVE. This working group standardized DNS over TLS

This working group standardized DNS over TLS in 2016 with RFC 2016. Since then Quad9 and Cloudflare being free, recursive, anycast DNS platforms supported DoT focusing on privacy protecting one part of the communication from the user to the resolver while the other part still communicated using clear text. However, in 2018 DPRIVE initiated the development of a solution to encrypt the channel between the resolver and name server.

## Research Overview

This document describes the usage of TLS to extend privacy for the Domain Name System. It also outlines security and privacy issues associated with the usage of DNS by Internet users. Although, this document does not provide solutions to the known DNS attacks, it highlights the need for encrypting the data and provides an overview of the existing situations and threats. By providing encryption, TLS removes chances for eavesdroppers to tamper DNS requests in the network query. Additionally, the document specifies two profiles for encrypting DNS queries: DNS over TLS (DoT), DNS over HTTPS (DoH) and provides analysis and review of performance considerations to reduce the overhead caused by using TLS and TCP over DNS. The document targets on improving security between stub and recursive traffic in par with the proposal of DPRIVE working group.

## Research Objective

Over the past months, I have been running a pilot in which I enabled DoT and DoH between a client and a recursive resolver. The aim is to interpret the feasibility of gathering metrics to better analyse the overhead suffered due to latency on collecting answers and measuring the overhead caused by computing. The described pilot will allow for better understanding of how the protocols performs. Furthermore, running this in a production environment will let us address issues that might arise while moving from DNS over User Datagram Protocol (UDP) to a set-and-forget approach to encrypted protocol such as TLS.

Until now, the pilot has proved that the performance of queries over TLS is similar to that of UDP for most test scenarios. Establishing the initial connection adds latency to the request however, reusing the TLS connection drastically reduces the latency for multiple queries. Thus, even though the initial overhead is high, it is on par and in some cases less than the UDP baseline.

The project will comprise of the following tasks on a high level.

* Exploring state-of-the-art perspective in evaluating the performance of DoT and DoH
* Review and contrast similar initiatives that consist of related work and compare inference with this project.
* Deploy and implement server and client software’s as well as download and setup certificates to setup a reliable connection between the stub and resolver.
* Analyse the packet using Wireshark and Tshark to gain valuable insight into the latency, packet size, loss and other metrics aiding in the comparison between UDP and TLS operations.
* Evaluation of outcome and exploring success and failure scenarios.
* Discussion of limitations, conclusions and future work

## Methodology Overview

To execute the objective of the research along with discussion on related work, many implementations will be listed. The literature review will further consist of procedures which are used in this thesis.

A multitude of software’s were used to achieve the aim of the thesis both, for DoT and DoH. The software’s common to both implementations are Python (version 3.6) and Selenium to automate the DNS operations on a variety of domain names. Javascript, HTML and CSS to generate charts and visualizations. The domain names were obtained from ------, processed using Excel and Python programming.

To implement DNS over TLS, Stunnel, a proxy with functionality of adding TLS encryption was deployed on the server side. While Stubby, an application acting as a local DNS stub resolver was used on the client side. Stubby encrypts DNS requests between client and recursive resolver, improving privacy of the end user.

To implement DNS over HTTPS, the following were used for the setup:

Nginx – To handle SSL certificates

DoH-Server – Translates Wireformat between UDP and HTTP

BIND9 – Handles name resolution, caching and optionally ad blocking.

## Dissertation Structure

# CHAPTER 2

# STATE OF THE ART

## Introduction

## Research Motivation

Over the past many years, attackers have hijacked domain names to manipulate and redirect users to malicious sites. Consider for example, a user trying to access a service portal, redirected to a website that seems legitimate. This compromises the user’s credentials through an authentication form or sometimes card payment details. The Gandi incident in 2017 is an example of a large-scale distribution attack wherein, the attacker gained access to a huge number of domain names of which 751 websites were compromised including modification of name servers to alter traffic flow to malicious sites exploiting security weakness in many browsers. Transport Layer Security, a cryptographic protocol is used to secure the traffic flow and communication in the World Wide Web. Since most of DNS queries are transferred as clear text, anyone will be able to sniff and gather information. DNS was formerly functioning on UDP as a stateless mode which made it more liable to IP address spoofing leading to impersonating server and sending forged queries. Since UDP does not support many encryption fixes, it becomes complicated.

Established on the rationale that any deliberate or incidental quality interruption or modifications of services supplied by the critical infrastructure may have socio-economic inferences and thus should be avoided and more attention should be directed towards the following three areas. First, to assure that critical components are protected adequately and identifying vulnerabilities and single points of failure. The focus on the second area is to detect proactively possible incidents that may affect normal operations, generating alarms thus, reducing the impact. The emphasis on the third area is to effectively restore and recover services post root cause analysis.

Lately, attacks involving misusing of components and exploitation of services to disrupt targets have grown in diversity and frequency. As an instant consequence, the variance between the target and the attack method has been blurring. A large number of initiatives to develop strategies to reduce or prevent attackers to attain illegal ends are rapidly growing [5]

We focus on the first area where we utilise existing implementations and analyse the usage to understand if they can be deployed at scale with significantly less overhead. In this thesis study, this approach is explored and followed to show the change in research importance is helpful for improving security of Internet. Specifically, it depicts how user machines use the service to resolve names provided by DNS and act accordingly representing a promising outlook to deal with a variety of attacks effectively.

In the light of the above-mentioned considerations, the study and result reported in this document has two primary research motives. The first one is to learn and spread knowledge on the threats that arise from the perspective of DNS and bring to light the many exploits against DNS and by that produce an apparent effect on DNS traffic. To demonstrate DNS over a connectionless protocol such as UDP to list the fundamental security and privacy weaknesses that can be addressed through DNS queries over connection-oriented DNS. Being a single packet exchange, DNS results in vulnerabilities, constraints on policies and such problems elevate when used in new applications and marks as a great concern as the privacy and security around Internet grows. The second objective is to demonstrate the analysis and control DNS traffic that passes over the Internet which is a promising direction to improve and explore. Protection of DNS traffic is important since the hostnames contain rich information than the visible IP addresses and DNS requests reveal application data. These queries are highly progressively vulnerable with increased wireless networks and rise in third-party DNS implying that the request crosses multiple networks that at a high risk of eavesdropping.

### 

Securing the Internet by Analysing and Controlling DNS Traffic: Email Worm Detection and Mitigation. <https://pdfs.semanticscholar.org/9896/161e2f0b0ffacc9ebd7804868a84b8b7cec1.pdf>

The contribution made in this thesis is not protocol novelty but an evaluation of what is required to improve the security of the established protocols in current infrastructures. We assess that the performance costs are moderate, and experiments prove that the privacy and security benefits are real. With common conviction that it is mandatory to use connectionless DNS for acceptable performance, the study addresses the main obstacle to achieve DNS privacy. While this thesis determines a generalised approach to connection-like DNS, roughly all of which need some state at either ends. Additionally, we describe usage and alternative protocols where DNS encryption could be a necessity and suggest optimizations.

## Public Nature of DNS queries

DNS queries are transmitted unenrypted and have least confidentiality concern. DNS publicly maintains records about domain names and the corresponding IP addresses. However, the hosts accessing this data is assuredly non-public. But this information may easily be available to other entities like organizations, government etc. by eavesdropping. The data that is revealed could provide information like a person’s email, contact, chat and more.

### DNS requests

As part of privacy concerns, the source IP address and QNAME are two particularly important fields which may reveal information about the user’s actions. Consider an example of a query containing host part as www.security.com of type A and requesting MX records for the domain. These types of requests disclose transmission relation [3]. Due to the lack of privacy it becomes risky to introduce personally identifiable, sensitive data in DNS and becomes a foremost concern if MUA’s explore PGP keys.

### Network Channels

Like any other traffic and since DNS queries are made in clear text, the traffic may be available by eavesdroppers. The DNS traffic may flow through recursive and authoritative name servers and not just between the sender and the receiver hence the access points are wider in case the direct channel is not available to tap. Since the medium between recursive and stub resolver is not restricted by caching, it is the convenient surface to tap.

Related Work and Comparison

There has been minimum work carried out to provide privacy between a client and a server. DNSSEC presents response integrity by using cryptographic signatures on zones permitting end users to validate if the replies are right. However, DNSSEC fails to preserve privacy of request and response. Other work includes DNSCrypt, DNSCurve, IPSECA and Confidential DNS.

The DNS traffic may leak information about the activity of a user on the internet giving away details of the email services they are using or the software the user is running. Repressive regimes may record activities of citizens browsing the internet, for example, look for users who resolve the name of a VPN server by an external organization which may help dissidents exchange information with the outside world.

Considering the public nature of DNS, securing and testing the performance against a wide variety of metrics is not something that hasn’t been done previously. Although, the choice of measurements, implementation and methodologies for doing so may vary.

### Connection-Oriented DNS to Improve Privacy and Security

The research carried out by Zi Hu in this paper follows almost the same motive as the topic of this thesis. It highlights the benefits of connection-oriented DNS at moderate costs. The end to end latency i.e. the time between the client requesting and obtaining an answer in a cached environment is reasonably more than UDP requests. According to the study conducted as part of this thesis, we can observe that in a cached environment, connection-oriented requests take considerably less time compared to connectionless queries. The study shows that there is 9% latency increase in TLS compared with UDP tested between the stub and recursive resolver and a 22% increase between the recursive to authoritative. Reusing the connection after the first connection establishment results in latencies almost like UDP. Timeouts of 20s were observed at the authoritative server and 60s elsewhere. Connection reuse is maximum for servers, decreasing costs for client and server. With conventional timeouts of 20s and 60s at authoritative and elsewhere respectively, the test conducted by the authors show that a recursive resolver on an average handles 24k active connections using around 3.6GB of RAM.

Slide-91-dprive

# BACKGROUND

This section provides an overview of how Domain Name System, Transport Layer Security, Hyper Text Transfer Protocol Secure function. It will also outline the threats common to DNS in today’s world.

Protocols

User Datagram Protocol

Hyper Text Transfer Protocol Secure

Domain Name System

DNS resolves domain names to various records in a database globally distributed. A user makes a query to a server that provides a response of a few types of records. They have multiple components and are hierarchical. The DNS database has a common root server and thousands of servers. The DNS resolves host names to IP address called forward resolution and address to host name knows as inverse resolution. The DNS has emerged as an important element of Internet due its distributed characteristics and robustness in mapping human memorable name to numerical address. Since accessing resources using IP addresses is not an efficient method, DNS is densely relied on to retrieve addresses by using Fully Qualified Domain Name (FQDN).

Since when the DNS was designed, it has evolved from resolving domain names to IP addresses to replica choice in content delivery networks (CDN) and host integrity identification due to its key to value mapping and its lightweight nature.

Cryptographic Primitives

Introduction to Transport Layer Security

Design and Implementation

The server implementation was done on Google Console Cloud. The specification of the Virtual Machine is as follows

1. Ubuntu 18.04 LTS
2. 1 vCPU
3. 5.25 GB memory
4. Europe-west6-a Zone

Datasets

DNS over TLS

Presently, all DNS requests are sent unencrypted making them vulnerable to snooping by an attacker who has access to the network channel. Thus, IETF has specified privacy considerations for DNS [11].

Initiation of DNS over TLS is done by establishing a connection over a well-known port through which the server and client are expected to create, agree and negotiate a session to secure the network channel. Currently, very few servers support DNS over TLS and sometimes the well-known port might be blocked by firewall rules.

As an initial requirement, we need a recursive resolver. For this purpose, we use BIND at the server side (Google Cloud Box). Berkeley Internet Name Daemon (BIND) is one of the oldest and popular implementations of DNS on Internet. It BIND distribution contains server software, client software and tools required for querying the DNS and troubleshooting underlying problems. All information in this document involving DNS implementation is done using BIND. We edit the configuration file, named.local.conf and edit parameters to make BIND server listen on the loopback address and instruct it to listen on port 853 which is the current default experimental port for DNS over TLS connections.

At the client side, I used Stubby. Stubby is an application that operates as a local stub resolver and uses DNS over TLS. It encrypts the queries sent from the stub machine to a DNS resolver thus, increasing the user privacy. Stubby is developed by DNS Privacy organization under the getdns project [12]. Stubby is light weight and although in early stages of development, it is suitable for customisation according to user needs. It acts as a daemon, listening to the loopback address and directs all the outgoing traffic over TLS by selecting one of the upstream servers provided in its configuration file.

By default, BIND9 does not support TLS encryption on port 853 and provide TLS auth credentials unlike Unbound and Knot Resolver [13]. To provide DNS over TLS service using BIND9, we use Stunnel. Stunnel, an open source application is a tunnelling service to provide universal TLS/SSL. It provides secure encrypted channels that natively do not communicate via TLS or SSL. It uses public-key cryptography with X.509 certificates to secure the communication channel. To generate this, we use Openssl [14].

#Create X.509 public key self-signed certificate for client performing without authentication.

openssl genrsa -out dns.key 1024

openssl req -new -key dns.key -out dns.crt -x509

Create a configuration file named stunnel.conf and add the keys generated in the previous steps in the file as below.

[dns]

accept = 853

connect = 127.0.0.1:53

cert = dns.crt

key = dns.key

Ideally, we need to create a real X.509 CA but since this is experimental, we created the CA certificate our self.

openssl genrsa -out dns.key 1024

openssl req -new -key dns.key -out dns.req

openssl x509 -req -in dns.req -out dns.crt -CA ca.crt -CAkey ca.key -CAcreateserial

Add this to the stunnel configuration

CAfile = ca.crt

While using Stubby, we need to generate a key-pin for the recursive server that will be used as an upstream server in Stubby. To do this we compute a sha256 pin by executing the following command.

openssl rsa -in dns.key -outform der -pubout | openssl dgst -sha256 -binary | openssl enc -base64

Launch stunnel as a daemon by running the following command

stunnel stunnel.conf

Once this setup is done, we run stubby service from the client side and modify the resolvd service to listen on loopback address. Fire up Wireshark, packet analyser tool to test and view the encrypted DNS connections over port 853. We use Domain Information Groper (DIG), a network administration command line tool for querying DNS name servers.

DNS over HTTPS

DNS over HTTPS is making an uproar since when Firefox browser has started supporting it. Over recent years, there has never been a shortage of idea about how DNS queries can be secured, and this includes initiatives like DNSCurve, Confidential DNS, DNSCrypt, DNS over DLTS and as previously discussed, the most recent DNS over TLS. An apparent issue is that there are plenty of them which is why IETF backed DoT to get moving. The reason for choosing DoT was that its operation was like HTTPS using the same TLS protocol using 853 instead of 443.

IETF efforts are directed towards standardizing DNS requests and responses that are applicable for HTTPS, allowing an interoperable and approved system for DNS queries to resolve over secure connections using HTTP/2 protocol. It is an integration of multiple approaches like error codes, methods and semantics while maintaining the format and latency of query replies as compared to the traditional DNS address resolution over UDP protocol.

Currently, users pay ISP for a connection to the Internet. As part of DoH, users establish a secondary connection to public or locally deployed recursive servers implementing it to make browsing private.

In this effect, big companies are adopting DoH to a roadway to secure connections. Google’s public DNS system, 8.8.8.8 or 8.8.4.4 will be supporting DoH at some point of time in one of the most popular browsers, Chrome and currently supported in Android 9. Mozilla Firefox on the other hand has rolled out DoH as an optional setting that needs to be enabled manually calling them Trusted Recursive Resolver or TRR.

With these in place, if a government body requires to know the websites visited by users, they can no longer ask the ISP. Instead, in theory they would be doing the same with Cloudflare, Google and Mozilla. Companies like Cloudflare have previously stated that the DNS queries are logged for 24 hours only and has a roadmap to publicly prove it [6]. These statements if true are comparatively better when compared to ISPs in most countries that collect user data for up to a year.

DoH is used as a DNS recursive resolution by resolvers. Resolvers or DoH clients should be connected to a DoH server that hosts a request endpoint.

DNS over HTTPS is not natively supported in operating systems. Hence, additional software’s must be installed. The following are the three common usage scenarios.

1. DoH within an application - Browsers like Mozilla Firefox as discussed in Section --- have a build in implementation of DoH hence can accept requests omitting dependency on operating systems DNS process. A disadvantage in this approach is that the user is not informed when an application skips a DoH request which could be wither due to lack of DoH support or misconfigurations. However, Firefox provides users with 4 modes to control how the resolution is made.
   1. Mode 1 – Allows Firefox to pick the fastest resolution protocol (UDP or over TLS)
   2. Mode 2 – The first choice of resolution is always over HTTPS and use traditional DNS as a fallback mechanism.
   3. Mode 3 – Strictly only encrypted queries over HTTPS
   4. Mode 0 – Default resolution over unencrypted channel
2. DoH proxy server on the name server in local network – In this context, client machines use regular DNS over port 53 or 853 to request a name server from the same network which then gathers required responses via DoH through DoH-servers on the internet. This ensures transparency to the users.
3. DoH proxy on local system - In this scenario, a locally deployed DoH proxy is queried by the operating system. In contrast to the previous method, the proxy should be deployed on every machine wanting to use DoH. This requires huge efforts if it is to be implemented in a large infrastructure.

In all the aforementioned alternatives, the client will not query authoritative servers instead, relies on DoH servers using port 53 or 853 to reach them. This makes a hop-to-hop encrypted channel only when DNS over TLS is used constantly. Thus, DoH does not qualify as an end-to-end encrypted protocol [7].

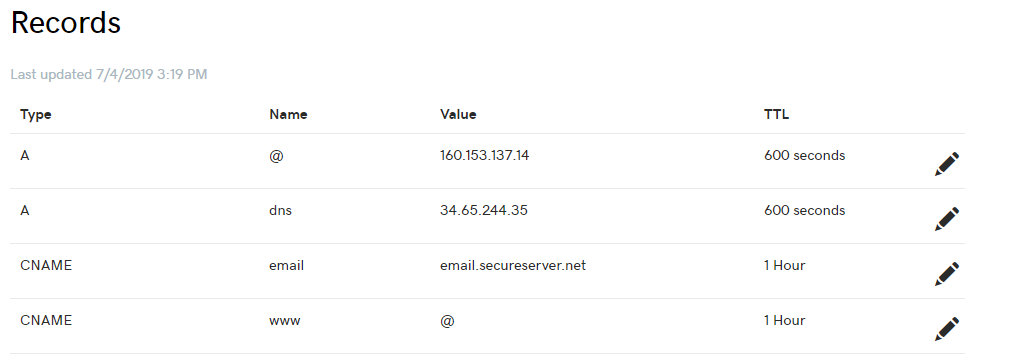
In order to setup DoH, I used doh-server which as the name suggests is a server that implements the protocol to receive a HTTP request and make a DNS query. In order to do this, I first installed Go, latest version 1.10 and compiled it. By default, doh-server uses Google DNS over HTTPS. This is changed in the configuration file. We change the upstream variable to listen to the loopback address on port 8053 of the server. For doh-server to successfully work, we need to install and configure Nginx to handle the HTTPS part of DNS over HTTPS. In order to do this, we configure Nginx as a reverse proxy and use Let’s Encrypt to create certificates.

Let’s Encrypt is a non profit organization whose main aim is to provide a simple way of setting up a HTTPS server and allow it to obtain a browser trusted certificate automatically without the intervention of a human. This is made possible by a certificate management agent running on the web server [2].

The certificate is provided in a two-step process. First, the Let’s Encrypt agent installed on the server side proves to the Certificate Authority (CA) that the webserver controls the domain in question by generating a new key pair similar to the process followed by the CA traditionally. The Let’s Encrypt CA looks at the website and issues a set of challenges. Apart from the challenges, the CA supplies a nonce that is signed by the agent’s private key to prove its control. An example of a task could be to create a file at a specific path on the domain. After completing the steps, the CA is notified which then proceeds towards validation and verifies if the expected content is present. To obtain the certificate, the agent composes a Public Key Cryptography Standards (PKCS) #10 Certificate Signing Request (CSR) requesting for a X.509 certification [9] from the CA that in turn asks Let’s Encrypt CA to provide a certificate for the domain with the public key specified. Second, the agent is responsible to request, renew and revoke certificates for the website.

To achieve all the above, OpenSSL, an open source software package is used for creating certificates. Let’s Encrypt scripts use OpenSSL to generate certificates, signed by the Let’s Encrypt service.

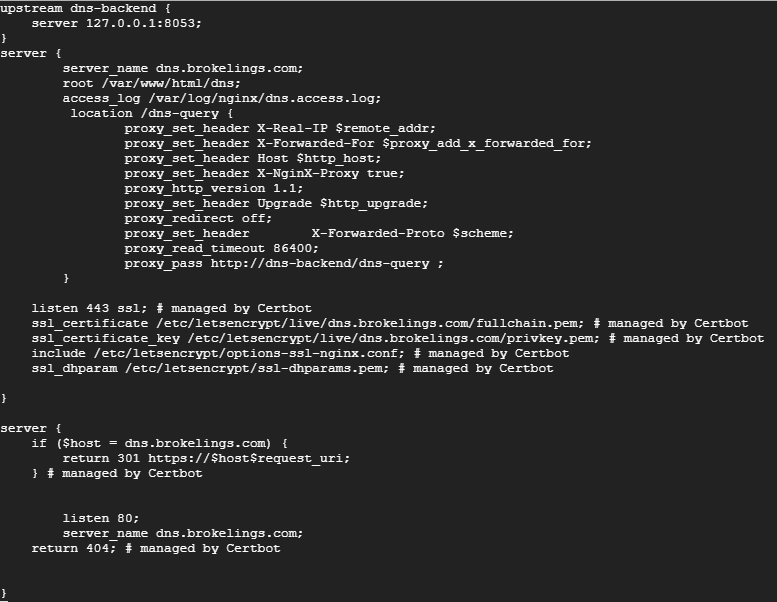
Once we have this, we get back to Nginx service and provide the location of the public, private keys and CA certificates. We purchased a domain name “brokelings.com” from GoDaddy (domain registrar), defined a DNS record “dns.brokelings.com” and created certifications from Let’s Encrypt on this domain name.



The server\_name field in Nginx file is changed to the domain name dns.brokelings.com used for DoH. The upstream\_server points to the machines loop back address and port where DoH is configured. We then enable SSL after following the procedure to create the required keys with Certbot and Let’s Encrypt. The content of the configuration file is placed in the location /etc/nginx/conf.d/\*.conf from where Nginx service will read from. We provide a symbolic link to the enabled folder and execute the following commands to check if the configuration works. Once the tests pass, we reload the Nginx service and we now have Nginx taking care of serving HTTP server requests to doh-server.

1. sudo ln -s /etc/nginx/sites-available/dns-over-https /etc/nginx/sites-enabled/dns-over-https
2. sudo nginx -t
3. sudo systemctl reload nginx

<Should I include this or no?>



vim /etc/nginx/conf.d/\*.conf

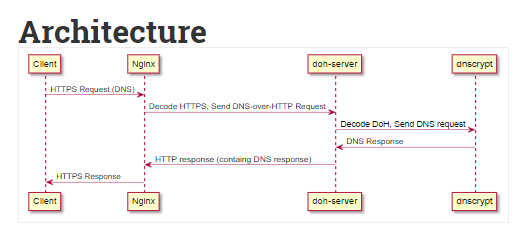
<Attach steps/commands of setting it up>

Currently only Mozilla Firefox provides support to DoH and can be enabled by following the below steps:

1. Type about:config in the address space of the browser and accept the risk by clicking on the button.
2. In the search bar, type trr to find the network.trr section of the settings.
3. The following fields are modified
   1. Network.trr.uri – In our case dns.brokelings.com/dns-query for which we have established doh-server to enable security.
   2. Network.trr.bootstrapAddress – Provide the IP address of dns.brokelings.com/dns-query so the browser can resolve network.trr.uri.
   3. Network.trr.mode – An integer between 0-3. In our case 3 to enable strict DoH.

We can test if the browser is picking up our configuration changes by visiting <http://www.whatsmydnsserver.com/> [10]. Presently, there are only 2 DoH publicly available servers namely, Cloudflare (https://mozilla.cloudflare-dns.comdns-query) and Google (<https://dns.google.com/experimental>). The later being an experimental server bound to fail occasionally.

Architecture



<https://www.aaflalo.me/2018/10/tutorial-setup-dns-over-https-server/>

Change of approach – write, why?

Connection Reuse

Future Work

Concurrent Connection

<https://blog.rapid7.com/2019/02/06/secure-that-query-researching-the-landscape-of-dns-over-transport-layer-security-tls/> - Contains some insights on which country, organization, provider use TLS the most

<https://code.fb.com/security/dns-over-tls/> - Facebook tests

<https://www.netsparker.com/blog/web-security/pros-cons-dns-over-https/> - Pros and Cons of DoH

<https://www.infosecurity-magazine.com/opinions/the-pros-and-cons-of-dns-encryption/> - Discussion on DoT performance

# REFERENCES

[1] European Commission. 2018 reform of eu data protection rules. https://ec.europa.eu/ commission/priorities/justice-andfundamental-rights/data-protection/2018- reform-eu-data-protection-rules en, 2018. [Online; accessed June-1-2018].

[2] Let’s Encrypt. <https://letsencrypt.org/>

[3] Quad9. Quad9 frequently asked questions. https: //www.quad9.net/faq/#What does Quad9 log/ store about the DNS queries, 2018. [Online; accessed July-4-2018].

[4] DNS over TLS: Encrypting DNS end-to-end. <https://code.fb.com/security/dns-over-tls/>. [Online; accessed July-31-2019]

[5] J. Englot and T. Zoli. Critical issues in achieving a resilient transportation infrastructure. In Symposium on Emerging Developments in Multi-Hazard Engineering, 2007.

[6] DNS over HTTPS is coming whether ISPs and government like it or not - <https://nakedsecurity.sophos.com/2019/04/24/dns-over-https-is-coming-whether-isps-and-governments-like-it-or-not/> [April-24-2019]

[7] DNS over HTTPS. <https://en.wikipedia.org/wiki/DNS_over_HTTPS>

[8] <https://github.com/m13253/dns-over-https>

[9] Wikipedia PKCS. <https://en.wikipedia.org/wiki/PKCS>

[10] Enable DNS over HTTPS in Mozilla Firefox <https://www.trishtech.com/2018/08/how-to-enable-dns-over-https-doh-in-mozilla-firefox/>

[11] Internet Engineering Task Force, S Bortzmeyer, sDNS Privacy Considerations - <https://tools.ietf.org/html/rfc7626>

[12] DNS Privacy Daemon – Stubby, DNS Privacy Project. <https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+Daemon+-+Stubby>

[13] DNS Privacy Implementation Status. <https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+Implementation+Status>

[14] DNS over TLS service using BIND9 and Stunnel. <https://kb.isc.org/docs/aa-01386>