BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

BIG DATA ANALYSIS TECHNIQUES FOR NETWORK TRAFFIC MONITORING: THE STORY OF DNS OVER HTTPS DETECTION

TECHNIKY ANALÝZY VELKÝCH OBJEMŮ DAT PRO MONITOROVÁNÍ SÍŤOVÉHO PROVOZU: PŘÍBĚH DETEKCE DNS OVER HTTPS

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR
AUTOR PRÁCE
KAMIL JEŘÁBEK

SUPERVISOR
ŠKOLITEL
doc. Ing. ONDŘEJ RYŠAVÝ, Ph.D.

BRNO 2023

Abstract

Network monitoring plays a crucial role in the arsenal of tools used by network operators to ensure security. With the majority of network traffic now encrypted and the emergence of new protocols that extend encryption to previously unencrypted communications, traditional monitoring techniques that rely on the visibility of unencrypted network traffic have become obsolete. Consequently, solutions must now depend on the traffic metadata provided by widely used flow monitoring infrastructures. One of the protocols that get encrypted alternatives is DNS. DNS over HTTPS (DoH) is one of the attempts to encrypt DNS traffic that received broad adoption among users and resolvers. The DoH implementation is already incorporated in most browsers, proxies, and operating systems. While DoH improves users' privacy, it leaves network operators and specialized Intrusion Detection Systems (IDS) blind to DNS traffic. Moreover, operators are unaware of DoH usage by users as DoH is designed to blend with other HTTPS traffic. Since its standardization in October 2018, the DoH has been studied extensively from various perspectives, including detection. This work proposes a reliable detection method using a combination of techniques, including machine learning, to identify DoH and distinguish it from regular HTTPS traffic, bringing awareness to network operators and allowing them to act according to their security policies. The work studies DoH thoroughly aligned with the data-centric concept of machine learning, enabling the creation of comprehensive datasets and designing effective practical detection mechanisms utilizing data sources of broadly present flow monitoring infrastructures. Moreover, the proposed detection method is tested in various scenarios, uncovering its characteristics and effectiveness compared with other state-of-the-art approaches.

Keywords

DNS over HTTPS, Network Monitoring, Detection, Machine Learning, Data Centric Concept, Data Analysis, Cybersecurity

Abstrakt

Síťový monitoring hraje klíčovou roli v arzenálu nástrojů používaných síťovými operátory k zajištění bezpečnosti. S většinou dnes již šifrovaného síťového provozu a s nástupem nových protokolů, které rozšiřují šifrování na dříve nešifrovanou komunikaci, se tradiční monitorovací techniky, které spoléhají na viditelnost nešifrovaného síťového provozu, staly zastaralými. V tomto důsledku musí řešení nyní spoléhat na metadata extrahovaná široce rozšířenými monitorovacími infrastrukturami pracujícími na úrovni síťových toků. Jedním z protokolů, který dostává šifrované alternativy, je DNS. DNS over HTTPS (DoH) je jedním z pokusů o šifrování DNS provozu, který získal širokou podporu mezi uživateli a překladači doménových jmen. Implementace DoH je již integrována do většiny prohlížečů, proxy serverů a operačních systémů. I když DoH zlepšuje soukromí uživatelů, zanechává síťové operátory a specializované systémy detekce vniknutí (IDS) slepé vůči DNS provozu. Navíc operátoři si nejsou vědomi používání DoH uživateli, protože DoH je navrženo tak, aby se zamíchalo mezi ostatní HTTPS provoz. Od standardizace v říjnu 2018 bylo DoH důkladně studováno z různých perspektiv, včetně detekce. Tato práce navrhuje spolehlivou metodu detekce s využitím kombinace technik, včetně strojového učení, k identifikaci DoH a jeho odlišení od běžného HTTPS provozu, což zvyšuje povědomí síťových operátorů o používání DoH a umožňuje jim jednat v souladu se svými bezpečnostními politikami. Práce podrobně zkoumá DoH v souladu s datově orientovaným konceptem strojového učení, což umožňuje vytvoření komplexních datových sad a návrh účinných praktických mechanismů detekce s využitím datových zdrojů široce rozšířených monitorovacích infrastruktur pracujících na úrovni síťových toků. Navíc navržená metoda detekce je testována v různých scénářích, odhalujících její charakteristiky a účinnost ve srovnání s jinými nejmodernějšími přístupy.

Klíčová slova

DNS over HTTPS, Monitoring sítí, Detekce, Strojové učení, Data centrický koncept, Datová analýza, Kyberbezpečnost

Reference

JEŘÁBEK, Kamil. *Big Data Analysis Techniques for Network Traffic Monitoring: The Story of DNS over HTTPS Detection*. Brno, 2023. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. Ing. Ondřej Ryšavý, Ph.D.

Rozšířený abstrakt

V současné době, firmy, organizace i jednotlivci čelí zvýšenému množství útoků. Velké množství útoků probíhá s využitím síťové komunikace. Již dlouhou dobu existují různá řešení pro zabezpečení této komunikace, která zahrnují systémy monitorování sítě, detekce vniknutí a prevence vniknutí. Nicméně tyto systémy stále alespoň z části spoléhají na inspekci nešifrovaného provozu. Avšak s postupným přechodem k šifrovaným protokolům, je většina dnešní komunikace již šifrována což snižuje využití předchozích technik a tedy omezuje některé předchozí schopnosti těchto systémů. Kvůli šifrovanému provozu jsme odkázáni na práci pouze s dosud nešifrovanými identifikačními částmi nebo statistickými metadaty jednotlivých paketů či celých spojení.

V posledních letech je zde snaha přivést šifrování i do dosud nešifrovaného protokolu pro rezoluci doménových jmen (DNS). Jedním z takových protokolů, který získal rychle na popularitě je DNS over HTTPS (DoH), kterému se tato práce věnuje. Od jeho schválené definice v říjnu 2018 se v krátké době stal součástí komunikace různých aplikací jako jsou webové prohlížeče, doménové proxy servery a dokonce operační systémy. Jeho rychlé adopci pohla kombinace s již existujícím HTTPS, kdy jsou DNS data nově přenášena tímto velmi rozšířeným protokolem. Nicméně, protože je DNS přenášeno pomocí HTTPS využívající stejný port transportní vrstvy jako zbytek HTTPS komunikace, je tento protokol z pohledu síťového monitoringu nerozlišitelný od ostatního HTTPS provozu. Tato technologie, která přináší větší soukromí uživatelům zároveň omezuje aplikaci současných bezpečnostních mechanismů založených na inspekci DNS provozu. Navíc, síťoví operátoři ztrácí schopnost identifikovat použití tohoto protokolu uživateli. Této vlastnosti pak využívají útočníci a škodlivé programy ke skrytí jejich aktivit.

Od definice tohoto protokolu se výzkumníci zabývají zkoumáním jeho charakteristik a návrhem metod jeho detekce. V posledních letech vzniklo několik takových metod. Současné navržené metody detekce jsou z valné většiny postaveny na využití strojového učení, kde vstupem jsou statistiky získané z paketů nebo síťových toků. Tradiční detekční metody založené na IP blocklistech se ukazují jako neefektivní vzhledem k velkému množství DoH serverů a jejich rychlému zastarávání a neúplnosti. Avšak tyto metody pracují se statistikami, které jsou mnohdy velmi specializované a obtížně získatelné z velkého množství toků, se kterými se současné monitorovací systémy musejí potýkat, což snižuje aplikovatelnost těchto metod v praxi. Tato práce si dává za cíl navrhnout novou detekční metodu založenou na strojovém učení, která je efektivní a použitelná v praxi.

Kvůli velkému množství dat, současné monitorovací systémy agregují pakety do síťových toků, ze kterých získávají omezené množství statistik, které je možné počítat efektivně a průběžně na běžící sekvenci paketů. Následně jsou exportovány pomocí standardních protokolů NetFlow a IPFIX. Pro zvýšení aplikovatelnosti navrhované detekční metody, se tato práce zaměřuje pouze na využití základních statistik extrahovaných těmito široce nasazenými systémy. Jelikož jsou dostupné vstupní informace poměrně výrazně omezeny, je nutné důkladně prozkoumat charakteristiky protokolu DoH. Mezi publikovanými pracemi zabývajícími se protokolem DoH existují mezery, které bylo nezbytné vyplnit pro navržení efektivní detekční metody, jako je zkoumání podoby DoH provozu a charakteristikami existujících implementací klientské i serverové části. Tato práce přináší výsledky těchto zkoumání. Navíc, existující publikované datové sady, se kterými výzkumníci pracují jsou omezené, nedostatečně variabilní, obsahující pouze malé množství různých serverů, jejichž implementace a konfigurace se ukázala mít největší vliv na vlastnosti DoH komunikace, nebo obsahují data pouze z uměle generovaného provozu.

Tato práce inklinuje k datově centrickému konceptu strojového učení, kdy hluboká doménová znalost vede k vytvoření dostatečně variabilních a obsáhlých datových sad, obsahujících i komunikaci z reálné sítě, které jsou součástí této práce, a které umožňují návrh výsledného detekčního řešení. I vysoce přesné metody detekce založené pouze na strojovém učení generují velké množství falešně pozitivních detekcí na síti v průběhu času. Navržená detekční metoda kombinuje několik přístupů včetně strojového učení a eliminuje počet falešných pozitiv. Metoda dosahuje vysoké přesnosti, vyšší nebo srovnatelné s ostatními publikovanými metodami a je ukázáno, že je méně náchylná k degradaci přesnosti v čase i při nasazení v jiném prostředí než ve kterém byla natrénována. Tyto vlastnosti jsou odhaleny a prezentovány ve srovnání s ostatními metodami, které se podařilo reprodukovat a validovat, na již zmíněné robustní kolekci datových sad, v závěru této práce.

Big Data Analysis Techniques for Network Traffic Monitoring: The Story of DNS over HTTPS Detection

Declaration

I hereby declare that this PhD thesis was prepared as an original work by the author under the supervision of doc. Ing. Ondřej Ryšavý, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Kamil Jeřábek
October 16, 2023

Acknowledgements

It has been a great journey since I started my Ph.D. studies in 2017. I have experienced a lot and matured in many areas, including article writing, team cooperation, networking, presentation skills, and many others. But it was not without sacrifice; the six years of my life devoted to my Ph.D. brought a few white hairs on my head. My studies also impacted my surroundings. Hereby, I want to thank all persons who helped and supported me during those wonderful but sometimes tough times.

First, I express my gratitude to my supervisor, Ondrej Rysavy, for his guidance, invaluable advice, and unwavering support throughout my entire Ph.D. study. One of the valuable things he gave me was granting me the freedom to chart my own course in research and self-motivation. He not only encouraged me to explore my ideas but also assisted me in turning them into reality. I am also deeply thankful to another distinguished member of our research group, Petr Matousek, for his insightful discussions, invaluable suggestions, and consistent support.

Our research group, NES@FIT, provided a welcoming environment filled with hardworking individuals, and I take immense pride in being a part of it. I owe a debt of gratitude to my colleague, Vladimir Vesely, who was a source of inspiration and motivation and who encouraged me during my master's studies to embark on this Ph.D. journey. I also appreciate my fellow colleagues, Libor Polcak and Matej Gregr, for engaging in discussions that often extended beyond the boundaries of my primary research focus.

A special acknowledgment goes to Jan Pluskal, a colleague and friend who shared an office with me, contributing to numerous enjoyable moments. I am also indebted to my other colleagues whose collective efforts created an inspiring and joyful atmosphere within the NES@FIT research group. They include Viliam Letavay, Michal Koutensky, Daniel Dolejska, Marcel Marek, Jan Zavrel, Martin Holkovic, Radek Hranicky, and many others. Additionally, I would like to express my gratitude to my former flatmate and colleague, Ondrej Lichtner, for his remarkable patience with me.

Another acknowledgment belongs to Dusan Kolar, the lead of the Department of Information Systems that our research group is part of, for his support in all my endeavors. I also extend my appreciation to our exceptional administrative professional, Michaela Bilkova, whose resourcefulness consistently ensured that tasks were completed. Equally deserving of recognition are the entire staff at our department and the Faculty of Information Technology for their collaboration and dedicated efforts.

I owe a debt of gratitude to my colleagues at the Czech Technical University and the CESNET organization, without whom completing this thesis, or at least doing so within such a brief timeframe, would have been considerably more challenging. Special thanks are in order for Karel Hynek, a colleague who worked diligently alongside me on research topics pertinent to this thesis, and his supervisor, Tomas Cejka, whose willingness to assist was greatly appreciated. Additionally, I am thankful to other colleagues and friends who made our collaboration productive and enjoyable, including Josef Koumar, Jaroslav Pesek, Richard Plny, Daniel Poliakov, and many others.

Throughout my academic journey, I had the privilege of spending a significant period at Strathmore University in Kenya. There, I not only met new colleagues but also forged enduring friendships. I extend my gratitude to Joseph Sevilla for his warm welcome at the institution, as well as to my friends Leonard Mabele, Diana Mutua, and John Ombagi, who introduced me to the diverse beauty of a different country and continent, making my stay truly special.

This entire Ph.D. journey would not have been possible without the unwavering support of those in my personal life. Thus, I would like to acknowledge the friends with whom I've shared these last years. Tomas Javurek, whom I met at the beginning of my studies, has since become a long-term friend and collaborator. I am thankful to Tomas for offering a unique perspective on various aspects of life, extending beyond the realm of art. Martin Janousek, another dear friend, encouraged me to not only focus on my Ph.D. studies but also engage in enjoyable, fun projects. His support was invaluable, particularly during the challenging times brought about by the COVID-19 pandemic. Engaging in discussions with Daniel Hecko about our mutual interest in Machine Learning broadened my horizons and led me to explore areas beyond the scope of my thesis. I would also like to extend my gratitude to Daniel and my friends Martin Chylek and Radomir Chylek for hosting memorable LAN parties, which provided me with a much-needed opportunity to unwind and relax. Jan Walczysko also helped me rest during our long discussions about various unrelated topics to my research, not only during our sailing trip. Tomas Hykel, who accompanied me during my stay in Kenya, added immeasurable joy to the experience. I am also grateful to my friends Jakub Mach and Marketa Babkova, who consistently provided a comfortable place to stay whenever I had the opportunity to visit Prague. I extend my thanks to Frantisek Kolacek and Jiri Prochazka for keeping me grounded through our insightful discussions about cybersecurity-related topics. My heartfelt appreciation goes out to all my other friends with whom I have had the privilege to share memorable moments.

I continue the rest of the acknowledgments in the Czech language since it is devoted to my closest family, and they would certainly appreciate it.

Nejvíce bych chtěl poděkovat mým rodičům Dušanovi a Šárce, kteří mě již od útlého dětství podporovali ve veškerém mém snažení. Rád bych jim poděkoval, za to, že vytvořili úžasné prostředí ve kterém jsem mohl vyrůstat a dosáhnout všech úspěchů mezi něž se počítá i sepsání této disertační práce. Taktéž bych rád poděkoval své sestře Markétě, se kterou jsem vyrůstal a zažil různá dobrodružství, a která mi vždy byla podporou. Naší fence rhodéského ridgebacka Abbygale Princess, která s námi vše v posledních letech sdílí, přináší radost a především za její vždy radostné přivítání. Vřelé díky patří babičce Věrce, se kterou jsme toho spoustu zažili a byla vždy nablízku. Pak také strýci Martinovi za ochotu a pomoc, neméně pak za organizaci rodinných událostí, při kterých jsme měli vždy příležitost se opět všichni setkat.

Rád bych také poděkoval své přítelkyni a partnerce Janě, jež se mnou prožila jak strasti tak i radostné chvíle v době mého Ph.D. studia a při psaní této disertační práce. Za to, že pomohla vytvořit láskyplné prostředí domova, že se mnou zdolala pandemii COVID-19 včetně jejích následků, a že mě vždy ve všem podporovala.

Poslední část poděkování již tradičně patří receptu na oblíbenou pochutinu, v mém případě se jedná o indickou čočku “Dal makhani”.

Ingredience: 1 hrnek černé čočky, 1/2 hrnku černých fazolí, 8 plátků čerstvého zázvoru, 5 lžic másla, 2 červené cibule, 5 stroužků česneku, 4 velká rajčata, 1 hrnek rajčatového protlaku, 1,5 lžičky garam masala, 1 lžička mletého koriandru, 1 lžička římského kmína, 250 ml kokosového mléka, sůl, pepř, čerstvý koriandr.

Postup: Přes noc namočíme čočku a fazole v trojnásobném množství vody se solí, plátky zázvoru. Druhý den vaříme, dokud se luštěniny neuvaří úplně do měkka, měla by vzniknout kašovitá hmota, voda se odpaří, někdy je třeba i přilévat. V druhém hrnci necháme na másle zesklovatět jemně nakrájenou cibulku, přidáme pasírovaný česnek a jakmile začne zlátnout, přimícháme všechno koření. Po pár minutách přidáme rajčata na kostičky a následně rozmixujeme tyčovým mixérem. Vše smícháme a za občasného míchání vaříme dalších 40 minut, můžeme vařit i déle. Na závěr vmícháme kokosové mléko, krátce prohřejeme, posypeme čerstvým koriandrem a můžeme podávat s rýží.

Contents

Chapter 1

Introduction

These days, companies are facing an increasing amount of attacks. From the network perspective, the security operators depend on network monitoring, intrusion detection, and prevention systems. The existing solutions rely on the ability of network content inspection. However, most of the network traffic is encrypted, and novel secure approaches to older unencrypted protocols are being proposed, decreasing the effectiveness of those systems. Moreover, the amount of data that needs to be processed is increasing and can be referred to as the Big Data problem. In combination with encrypted traffic, techniques such as deep packet inspection can no longer be used effectively. Therefore, the monitoring solutions rely more on network metadata extracted from aggregated information of network connections that can be extracted even on high-speed and backbone networks.

One of the network protocols that is still used unencrypted is the Domain Name System (DNS) protocol. The protocol is getting its encrypted alternatives, such as DNS over TLS (DoT), DNS over Quic (DoQ), and DNS over HTTPS (DoH). The motivation behind making the protocol encrypted is to improve the users' privacy and prevent potential observatories, such as Internet Service Providers (ISP) and other parties, from being able to spy on them.

The last mentioned protocol, DNS over HTTPS, was proposed five years ago and became very popular with its implementation immediately present in major browsers [8, 9, 115], later even in operating systems [59, 113, 132] and other applications. The protocol uses a simple principle where unencrypted DNS messages are transferred via an encrypted Hypertext Transfer Protocol Secure (HTTPS) channel. This simple solution makes it very easy to implement either on the client or server side, where just a combination of two existing solutions is required.

Nevertheless, this privacy-preserving technology raised concerns mainly from the security community [86, 137]. The security systems strongly rely on the inspection of plain-text DNS. Encrypted DNS bypasses the security solutions, which become unusable and blind, increasing the risk of potential security breaches. The main concerns about this particular protocol are attributed to its stealthiness. The DoH is designed to blend into other HTTPS traffic since it uses the same port as regular HTTPS protocol. This becomes very risky since the security operators are losing the ability to identify whether the protocol is being used on the network. In contrast, other mentioned encrypted DNS protocols operate on dedicated ports and can be easily identified by them. Moreover, due to this very critical ability (stealthiness), the protocol became very popular among the threat actors [62].

As mentioned, detecting DoH is not possible by simple techniques such as port identification. Moreover, the wide internet scans to build Internet Protocol (IP) based blocklists proved to be highly unreliable due to many private resolvers [44, 45]. Therefore, other techniques, such as machine learning-based detectors, provide a viable option to detect DoH traffic. This area of research represents an open challenge several research teams are competing in, which was also identified by Hynek et al. [62] in their survey.

Reliable identification of DoH in regular HTTPS traffic would be very beneficial to network operators and would help improve security. This work aims to research a solution, proposing a reliable and effective detection method and further studying the DoH behavior.

1.1 Goals of the Thesis

The main goals of this thesis come from the motivation of the work to research a reliable solution to detect DoH, as stated, that would be beneficial to the network operators and help increase security. The main goals of the thesis are the following:

- Propose a reliable detection method to identify DoH and distinguish it from other regular HTTPS traffic (covered in Chapter 7).
- Test the proposed method and challenge it in multiple scenarios, revealing its characteristics and usability in the real environment and further comparing it to other published methods (covered in Chapter 8).

The main goals would not be possible to achieve without a deep understanding of the protocol behavior. Moreover, since the work intends to use machine learning to solve the defined problem, quality data to work with is an essential asset. Hence, working on the main goals brought other goals and results that are not of less importance. The other goals are the following:

- Analysis of the DoH resolvers representing one side of the communication and their characteristics (covered in Chapter 4).
- Analysis of the impact of DoH on the browsers' behavior as the first and widely used stable client applications adopting resolution technology and single query characteristics (covered in Chapter 5).
- Analysis of the traffic shape of DoH (covered in Chapter 5).
- Creation of robust and comprehensive datasets as a source to support the creation of detection method and to provide a source of data for methods comparison (covered in Chapter 6).
- Further creation of supportive applications such as monitoring of resolvers and experimental PCAP processing tool NetExP (covered in Chapter 2).

Author's Contribution

The findings presented as author's work in this thesis were either independently created and assessed by this thesis's author or collaboratively produced within a team. When working in a team, the author was the leading person in a team responsible for the results. All the interim, as well as the final outcomes, were achieved with valuable assistance and under the supervision of doc. Ing. Ondrej Rysavy, Ph.D.

1.2 Thesis Structure

The thesis starts by providing necessary information about the technologies involved, described in Chapter 2. Current state-of-the-art is covered in following Chapter 3. The analysis starts with Chapter 4 that analyzes the well-known DoH servers as a representative sample that should be known to the users and from which the users might likely choose. Moreover, different server deployments may influence traffic characteristics the most. Chapter 5 aims to analyze the behavior of the DoH. It begins with looking at single query DoH (some applications such as malware might use) performance, then analyzing the impact of different HTTP methods on DoH in browsers, and then finishes with traffic shape analysis uncovering the DoH traffic characteristics. Chapter 7 proposes a reliable DoH detection method. Moreover, the detection method is challenged by several tests (covering testing on real-world traffic and data drift testing) as presented in Chapter 8. The detection method is also compared to other published methods that were thoroughly reproduced for this particular purpose. Finally, the work is concluded in Chapter 9.

Chapter 2

Background

This chapter covers the necessary technical background for the work. An introduction of the core DNS over HTTPS (DoH) protocol is provided. Since DoH is a composite of existing protocols, a brief description of the aspects and characteristics of the compound protocols influencing the behavior of DoH is also covered. Moreover, several other encryption protocols exist or are being proposed, and a short introduction is also provided to put the DoH into the context of DNS over Encryption. Network monitoring is an area of this work that provides a data source for the study, and the proposed solution relies on it. Moreover, the proposed solution is targeted to be deployed alongside an existing network monitoring infrastructure. The last area focuses on machine learning and data processing. The proposed solution is designed using several shallow learning algorithms; the data are preprocessed, and the results are benchmarked using several metrics. All those aspects are covered in this chapter.

2.1 Protocols

DoH is a compound of existing protocols, which includes unencrypted DNS protocol carrying the core queries and responses. DNS messages are transferred through an encrypted HTTPS channel that combines TLS and HTTP protocols. All of those protocols have their characteristics. Hence, this section briefly introduces them. Moreover, a brief introduction of other DNS over Encryption alternatives is provided to put DoH into a context of other DNS over Encryption alternatives.

2.1.1 Domain Name System

The Domain Name System provides a DNS protocol that enables hosts the translation of domain names into IP addresses. The protocol was first introduced in RFC 1035 [95] and extended by numerous RFCs later[1]. The communication uses messages of a simple format depicted in Figure 2.1.

The message is divided into multiple parts, where some are optional. The message always contains a header determining whether it is a question or an answer. The header is followed by the question part that carries three fields: query type, query class, and query domain name. This part is followed by a possibly empty list of resource records with the *Answer* that answers the question, the *Authority* that points toward an authoritative name server, *Additional* section contains additional information related to the query but not strictly answering the question itself [95].

The DNS message can be transmitted using both UDP and TCP transport protocols. The reserved port of DNS is 53 in both cases. The UDP payload messages are restricted to 512 bytes.

[1]https://help.dyn.com/articles/dns-rfcs/

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward an authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

Figure 2.1: DNS message format [95].

TC bit is set in the header for a longer DNS message, and the message is truncated. In the case of TCP protocol, the two-byte length field is added before the DNS message in the TCP payload, giving the length of the message [95]. A DNS message can use the maximum TCP payload size, which is not limited to only 512 bytes, as in the UDP case. Later, RFC 6891 [33] defines a way how to advertise an option to set a larger response size so that messages reaching more than 512 bytes does not need to be truncated and can be transmitted over UDP.

EDNS(0) Padding Option

The Extension Mechanism for DNS (EDNS(0) also denoted as EDNS in this work) Padding Option defined in RFC 7830 [89] is an option extending the DNS protocol. It can influence the traffic shape of the encrypted DNS. This feature adds padding of determined size to the DNS messages, both queries and responses. The queries and responses transmitted over encrypted channels can be subject to correlation analysis, where the sizes can be used to reconstruct the original queries, knowing their unencrypted counterparts. The padding in the encrypted DNS traffic aims to make it harder to apply such correlation techniques. Later, it was extended in RFC 8467 [90] by introducing several policies that can be applied.

2.1.2 HTTP(S)

HTTP is part of the DoH technical stack and exists in various versions. The HTTP/1.0, a former HTTP protocol, was closely followed by the HTTP/1.1, which introduced extensions to the protocol and improvements such as persistent connections, advanced caching options, compression, and so forth [43].

The HTTP/2 was introduced in 2015 by RFC 7540 [13]. HTTP/2 version contains many changes to the previous one. The former text-oriented protocol is now binary, contributing to its overall efficiency and extending the functionality that correlates better with today's needs. The protocol also allows multiple streams over a single TCP connection, prioritization, and multiplexing. In addition, the header and the data can be transmitted as separate parts in different data frames. The protocol is more complicated to implement than the previous versions, leaving more room for the variances in implementations and options used.

The TLS protocol provides a secured stream-oriented connection to HTTP. The HTTP protocols are mostly encrypted these days. HTTP/1.1 may be seen unencrypted, but HTTP/2 is used solely in combination with TLS despite the existence of HTTP/2 over TCP, which is mostly not implemented [111]. Several TLS versions exist, particularly TLS 1.0, TLS 1.1, TLS 1.2, and TLS 1.3. The first two versions are already deprecated, suffering from various attacks and providing weak encryption mechanisms. HTTP embedded in TLS is denoted as HTTPS. The TLS 1.3 [114] is the newest version currently on the rise while adding more security and improving connection efficiency, but an abundance of HTTPS traffic uses TLS1.2.

2.1.3 DNS over HTTPS

The DoH was standardized by IETF as RFC 8484 [51] in 2018 to enhance the privacy of domain name resolution and prevent on-path devices from interfering with the DNS resolution process [51]. The RFC-compliant DoH protocol uses traditional DNS wire format messages as defined in RFC 1035 [95] and sends them to the resolver as HTTP POST or GET requests. The DNS messages can be created as defined by DNS standard, including additional options and extensions such as EDNS(0) [89]. The DoH is designed and recommended to be used in cooperation with HTTP/2 [13] to maintain its efficiency due to the possibility of sending multiple concurrent requests, which overcame the head-of-line blocking on the application layer that was an issue in the case of previous DNS over TLS. However, the head-of-line blocking is still present on the transport layer. The protocol utilizes features such as creating multiple streams within a single TCP connection and multiplexing using separated frames to transmit headers and data.

DoH brings a disadvantage of an additional payload of HTTP headers. To identify DNS data in the HTTP stream, string *application/dns-message* should be used for *Content-Type* header in all DoH messages and in *Accepts* header field for queries. RFC 8484 does not define any additional specialties that should be included in the application messages except for new headers, possible cache-control options, and the body data format. However, the amount of headers is not limited, and DoH clients and servers are free to include more headers in DoH messages as necessary. In addition, the DoH is defined to operate on HTTP path *dns-query*, but not strictly limited to. In reality, many different URL paths can exist.

Compared to other DNS protocols, DoH does not have a specially dedicated transport port but uses HTTPS port 443 along with regular HTTPS traffic. Since a specific URL path must be used to target specific DoH resolvers, many different DoH resolvers can reside on the same domain, sharing many IP addresses. Moreover, the DoH is meant to be used solely between clients and resolvers, not encrypting communication between the resolvers.

POST Method

HTTP POST method is one of two HTTP methods that are defined in RFC 8484 to be used for handling DoH queries. Figure 2.2 depicts HTTP POST carrying DNS example query. The HTTP header contains the request field *Content-Type* representing the media type of value *application/dns-message*. The body of the message contains a DNS query in the DNS wire format [95]. The HTTP header may contain *Accept* field to clarify what type of response the client expects, e.g., binary.

GET Method

The second defined option to carry DNS queries is using the HTTP GET method. DNS query transmitted via HTTP GET is shown in Figure 2.3. The request's body is empty as the queried domain is encoded in *base64url* format as the value of the *dns* variable.

DoH Response

The DoH response (see example in Figure 2.4) should be of type *application/dns-message*, though RFC 8484 [51] notes that other formats may be introduced. The data payload for this media type is a

```
:method = POST
:scheme = https
:authority = dnsserver.example.net
:path = /dns-query
accept = application/dns-message
content-type = application/dns-message
content-length = 33

<33 bytes represented by the following hex encoding>
00 00 01 00 00 01 00 00  00 00 00 00 03 77 77 77
07 65 78 61 6d 70 6c 65  03 63 6f 6d 00 00 01 00
01
```

Figure 2.2: DOH POST query example [51].

```
:method = GET
:scheme = https
:authority = dnsserver.example.net
:path = /dns-query? (no space or Carriage Return (CR))
        dns=AAABAAABAAAAAAAAAWE-NjJjaGFyYWN0ZXJsYWJl (no space or CR)
        bC1tYWtlcy1iYXNlNjR1cmwtZGlzdGluY3QtZnJvbS1z (no space or CR)
        dGFuZGFyZC1iYXNlNjQHZXhhbXBsZQNjb20AAAEAAQ
accept = application/dns-message
```

Figure 2.3: DOH GET query example [51].

single message in the on-the-wire format. The maximum DNS message size is defined to be 65535 bytes, but in reality, the average size of each message payload reaches about a hundred bytes [42].

Each DNS request has its counterpart response mapped to one HTTPS message exchange. When combined with HTTP/2, the messages can be transferred over multiple channels, allowing out-of-order delivery of particular responses thanks to HTTP/2 multistreaming functionality.

```
:status = 200
content-type = application/dns-message
content-length = 61
cache-control = max-age=3709

<61 bytes represented by the following hex encoding>
00 00 81 80 00 01 00 01  00 00 00 00 03 77 77 77
07 65 78 61 6d 70 6c 65  03 63 6f 6d 00 00 1c 00
01 c0 0c 00 1c 00 01 00  00 0e 7d 00 10 20 01 0d
b8 ab cd 00 12 00 01 00  02 00 03 00 04
```

Figure 2.4: DOH response example [51].

DNS over JSON

Another DoH option can be considered DNS in JSON defined in RFC 8427 [49] and is worth mentioning since it is also meant to be transmitted over HTTPS. Global DNS resolver providers such as Cloudflare and Google implement the option to exchange DNS information using JSON format. It can be considered as a regular REST API call returning JSON data object. In this case, the client set *content-type* to *application/dns-json*.

The request URI can be seen in Figure 2.5. The main variables are *name* for the domain name in plain text and *type* for the DNS response. The response can be seen in Figure 2.6. The response fields and information are similar for both Google and Cloud Flare providers (provided examples come from Cloud Flare developers' documentation [29]).

```
curl -H 'accept: application-json' \
'https://cloudflare-dns.com/dns-query?name=example.com&type=AAAA'
```

Figure 2.5: DOH JSON request example.

```
{
  "Status": 0,
  "TC": false,
  "RD": true,
  "RA": true,
  "AD": true,
  "CD": false,
  "Question": [
    {
      "name": "example.com.",
      "type": 28
  ],
  "Answer": [
    {
      "name": "example.com.",
      "type": 28,
      "TTL": 1726,
      "data": "2606:2800:220:1:248:1893:25c8:1946"
    }
  ]
}
```

Figure 2.6: DOH JSON response.

This particular method is not considered to be covered in this work and is not further referred to as DNS over HTTPS.

2.1.4 Current State of DoH Implementation

The DoH is a protocol defined a few years ago and drafted in RFC 8484 by P. Hofmann (from ICANN) and P. McManus (from Mozilla) [50]. The publication of this RFC unleashed a wave of interest as well as criticisms.

The effort of content delivery network providers, in combination with application and operating system vendors, accelerated the adoption of the DoH protocol. Mozilla added the first support for DoH in version 62 of their Firefox browser [115], and in February 2020, DoH was the default option for all US users [36]. Chromium project introduced support for DoH in version 83 of Chrome browser [9]. Support in other browsers based on the Chromium code base, such as new Edge, Brave, and Opera, followed shortly. Furthermore, Chrome for Android devices introduced the support since version 85 [8]. Microsoft enabled DoH in Windows 10 operation system insider preview build 19628 by the 13th of May 2020 [79]. Apple also announced the DoH for iOS version 14 and macOS version 11 [25]. Moreover, the number of DoT/DoH servers has increased significantly in 2019, as Chaoyin Lu et al. [83] reveals.

2.1.5 Other DNS over Encryption Protocols

Other DNS over Encryption protocols providing alternatives to DoH include DNS over TLS (DoT) standardized in 2016, and the most recent standard, DNS over QUIC (DoQ) standardized in May 2022.

The DNS over TLS protocol brings encryption and authentication to the DNS communication by encapsulating the DNS messages directly in the TLS session. It is defined in the RFC 7858 [57]. Standard port 853 is used as the default port for this protocol. The communication between two entities, client and server, uses the TCP transport protocol. Once the client creates a successful TCP connection, it proceeds with TLS Handshake defined in RFC 5246 [37]. The authentication continues if required, but it is not necessary. To minimize the latency, the clients do not have to wait for the responses, and they can pipeline multiple queries over one TLS session. The transmitted data are the same as in the case of DNS used with TCP transport protocol. Compared to DoH, which is meant to be used with HTTP/2, DoT has the disadvantage of head-of-line blocking on the application level [14]. The client has to wait for responses to the sent queries, which is effectively overcome by the HTTP/2 multistream feature in DNS over HTTPS.

The DNS over QUIC brings another encryption alternative for DNS communication. The DoQ was finally standardized in RFC 9250 [60], but the beginning of the proposals can be traced down to 2017. Compared to the DoH and DoT, it operates over UDP transport protocol instead of over TCP. The DoQ uses QUIC [63] protocol directly to transmit the DNS messages. The QUIC protocol is not affected by the TCP packet loss and retransmission, where the communication needs to wait. Thanks to multiple independent data streams, it overcame the head-of-line blocking either on the application level and transport layer. The protocol is designed not only to transmit the DNS messages between clients and resolvers but also between resolvers. The DoQ has defined dedicated port 853 similarly to DoT but on UDP. The DoQ excludes the need for HTTP to be part of the communication, increasing privacy and reducing unnecessary payload to be transmitted. Due to the existence of several DoQ drafts, the implementation is ongoing, and many operating resolvers exist with various draft version implementations, as shown by Kosek et al. [76].

The two aforementioned protocols provide a viable alternative to the DoH with different advantages and disadvantages. However, both have defined dedicated ports anchored in the RFCs definition and can be easily identified or blocked.

2.2 Network Monitoring

Computer network monitoring and network traffic processing work with the logical assets of the network traffic, the packets. The packets are sequences of bits that together create one small fragment of a logical piece of information transferred over the network.

Full packet capture is one of the ways to deal with network traffic. Many network monitoring and processing tools work directly with the packets, extracting and analyzing information from them. Techniques such as deep packet inspection can be used, and all layers can be traversed up to the application layer where the user data of interest resides. Many tools focusing on this task exist, either open-source or commercial. However, as almost all network traffic became encrypted and the data are no longer readable by observers, the usability of such tools is decreasing. Moreover, the amount of data that is transferred over the network is increasing, increasing the cost and required processing power of solutions, further limiting the usability. The packet-level tools mostly focus on the rest of the unencrypted protocols, such as DNS. They are intended for offline full packet capture processing and forensics. This extreme approach of full packet capture benefits from information from all packets available.

Flow-based monitoring is an alternative widely adopted these days and already present on most network devices, such as switches and routers. The flow-based monitoring represents a compromise where the packets belonging to one connection are aggregated into so-called flows. Only metadata such as packet lengths and timings are mostly collected per flow. The flow monitoring and processing tools rely on such metadata and work solely with them. Compared to full packet capture, some data are lost, but when most protocols are encrypted, the loss is not that significant. Moreover, this approach can be used on high-speed and backbone networks. It can be considered as a Big Data approach. As many works proved, the aggregated flow data can be reliably used in combination with detection methods for various attacks [93, 97] and are utilized by researchers to solve various network-related tasks [123]. Moreover, many commercial solutions are based on network flows data source, e.g., Flowmon ADS[2], Cisco Stealthwatch[3].

This work consider only flow-based approach, since most current monitoring infrastructure support it which makes it viable data source for practical system. Special emphasis is on the usability of the proposed solution that can be deployed to real, even high-speed networks.

2.2.1 Flow Monitoring Architecture

As mentioned earlier, the flow-based network monitoring approach is one of the prevalent approaches [53], mainly due to its flexibility and broad support by networking hardware [125]. The typical flow monitoring infrastructure diagram is provided in Figure 2.7.

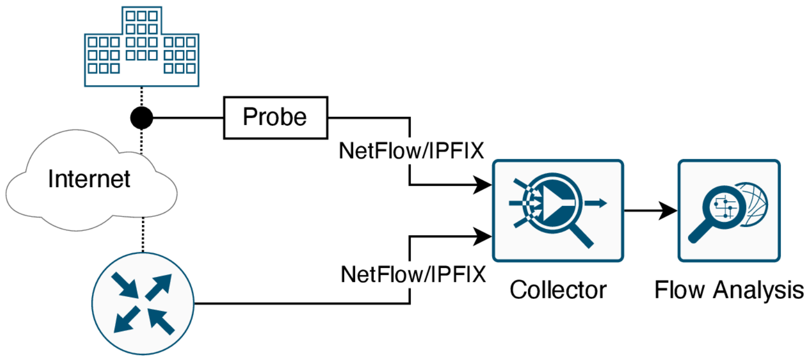


Figure 2.7: Simplified diagram of flow monitoring infrastructure.

The flow monitoring infrastructure consists of network probes or network devices capable of flow export distributed throughout the monitored network. The probes observe ongoing traffic and create statistical flow records, which are then sent to a single collector device to perform the analysis and post-processing.

2.2.2 Network Flow Representation

The data extracted by monitoring probes or network devices that the collector stores are flow records. Flow records can be defined as aggregated information of packets transferred through the network that shares some common properties [53]. The common properties are called flow-key and usually consist of IP addresses, ports, and transport protocol.

Besides the flow key, there are no standardized features that each flow record needs to carry. Flow traditionally contains the number of transferred bytes and packets [85]. These standard flow features

[2]https://www.flowmon.com/en/products/software-modules/anomaly-detection-system
[3]https://www.cisco.com/c/en_hk/products/security/stealthwatch/index.html

can be extended for information from an application layer (such as HTTP headers, DNS payload, or Transport Layer Security fingerprints) or statistics of the aggregated communication. However, these extended flow fields are not standard and vary widely based on monitoring infrastructure [53]. Most of the probes work with statistics that can be computed on running sequences and have low space and time complexity.

Table 2.1: Flow record features available across multiple Flow Export protocols.

Flow Record
Source IP address
Destination IP address
Source Port
Destination Port
Transport Layer protocol
Number of packets
Number of bytes
Time start
Time end

The flows are transmitted between probes and collectors using flow export protocols. Multiple standardized flow export protocols are currently in use [53]—NetFlowV5 [27], NetFlowV9 [28], and IPFIX [4]. While NetFlowV9 and IPFIX support templates, and thus it is possible to customize the flow features highly, NetFlowV5 has fixed records and does not allow any addition of flow features [27]. NetFlowV5 thus highly limits the feature availability. However, according to Hofstede et al. [53], it is still the most used flow export protocol. Table 2.1 shows a subset of features carrying identification and flow statistics that are commonly present [53], and are supported by all currently used flow export protocols.

According to Hofstede et al. [53], flow records usually describe only unidirectional communication. Thus, the bidirectional network communication is split into two records, aggregating packets in a single direction. When necessary for additional analysis, the bidirectional flows can be created from two unidirectional by applying flow stitching on the collector.

2.2.3 Network Flow Traffic Processing

Despite the existence of commercial and open-source monitoring tools such as exporters or probes, there is a need to explore new useful attributes that can be extracted from flows and used for machine learning. Existing flow records can be extended by the new attributes if necessary. Several such tools are available and provide an opportunity to process offline traffic with which the researchers work. This is especially important as it allows the opportunity to analyze traffic and extract characteristics that can be used for detection using machine learning techniques.

One such popular open-source tool is a *Wireshark*[4] and its command line alternative *tshark*. The *tshark* can process network traffic in the form of raw PCAP files and aggregate packets into connections. It provides many parsers so it can precisely parse various application protocols. However, its ability to extract custom statistics from flows is limited or complicated.

One of the big data tools that can extract packet data from raw PCAP files and aggregate them into flows with statistical information extracted is *Spark-ndx* processing platform built on Apache Spark, extracting data stored in Hadoop Distributed File System (HDFS), and results are pushed into

[4]https://github.com/wireshark/wireshark

Cassandra database. It should be deployed on distributed hardware and can process vast amounts of data as measured in Jerabek et al. [67] work. However, the need to operate such a platform on distributed hardware makes it very impractical for researchers.

The *nfdump*[5] represents a known open-source toolset that can work with Netflow data supporting various formats, including NetflowV5 and IPFIX. The tools can even process raw PCAP files and extract Netflow data directly from them. Despite the many options to configure and fast processing, the toolset provides limited or complicated options to craft custom statistics extracted from the traffic.

The *ipfixprobe*[6] is another open-source tool developed by CESNET organization that processes raw traffic or offline PCAP stored traffic and extracts IPFIX records. The tool is fast and low resource intensive but with limited configuration options extracting only predefined features. Additionally, the tool provides values such as the payload size of a sequence of the first 30 packets that can later be used for ad-hoc custom statistics computing.

The *DoHLyzer*[7] or similar *CICFlowMeter*[8] are tools for extracting network flows from offline captured PCAP traffic. The tools extract several statistical attributes from bidirectional flows. The tools are slow to process and very resource-intensive, with few options to configure and extract predefined features only.

Many tools can extract aggregated flow data with statistical features from captured network traffic or an interface. However, the tools are either complicated to maintain, resource-intensive, slow, or lack configurability or easy extensibility. The machine learning discipline relies on data, and as mentioned earlier, the data can be transformed and aggregated in a different way to gain new attributes, providing better knowledge. Therefore, the *NetExP*[9] tool that represents an easily extensible framework for flow aggregation and statistical feature extraction was developed. This tool is further used in this work as a main input data source.

2.3 Machine Learning Methods

The work involves machine learning and data mining methods to propose a detection solution. This section briefly describes methods, machine learning algorithms, and metrics used for performance benchmarking in this work.

In the case of this work, the data source is the network flow described earlier. The IP source and destination address is an identifier of packets at the IP layer. The source and destination ports are identifiers of the transport layer. Another important property is a transport protocol. These attributes create a 5-tuple, the identifier, for each flow.

Besides the 5-tuple identifying flow, each flow is further represented as a tuple (vector) of attribute values $V = (x_1, x_2, ..., x_n)$ where n is a number of attributes. Each x_i represents one of the statistical attributes calculated on a single flow. The attributes describe the characteristics of flow. A. W. Moore et al. provided 248 flow discriminators for the flow classification in [98]. Not all attributes are needed to produce reliable classifiers. The correct amount and combination of attributes should be researched with respect to other aspects, such as the computational complexity of attributes. A tuple consisting of attribute values is then used for further processing and machine learning.

[5]https://github.com/phaag/nfdump
[6]https://github.com/CESNET/ipfixprobe
[7]https://github.com/ahlashkari/DoHLyzer
[8]https://github.com/datthinh1801/cicflowmeter
[9]https://github.com/kjerabek/netexp

2.3.1 Data Preprocessing

The data preprocessing includes several data transformation techniques that either transform the input data for better understanding or for easier algorithm handling. Some of those techniques were used in this work.

Aggregation

One of the data transformation techniques is aggregation. The aggregation takes pieces of data that logically belong together (individual packets in this work) and aggregates them into flows. Then aggregated information such as sums or means of new aggregated entity is computed.

Attribute Construction

New attributes can be constructed from existing attributes by using arithmetic operations and added to the current set of attributes. The newly constructed attributes can help represent the data better. Such a process is also known as feature polynomialization.

Normalization

Normalization (also called scaling) is a technique that transforms data so that it falls into a certain range, e.g., between 0 and 1. This input values transformation technique is especially useful for neural networks, or algorithms involving distance measuring such as K-nearest neighbors, or clustering. While for other algorithms, such as decision trees, the technique does not bring any benefits as it is invariant [47]. Two commonly used methods are *min-max* and *z-score* normalization.

The *min-max* normalization performs a linear transformation of values of a certain attribute. Suppose that all values of an attribute A fit in between min_A and max_A. Then the normalization maps the $value_A$ to new_value_A so that it fits the range of new_min_A and new_max_A. The transformation function looks as follows:

$$new\_value_A = \frac{value_A - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A \tag{2.1}$$

The commonly used min-max normalization transforms values to fit between 0 and 1.

The *z-score* normalization (also called standard or zero-mean) is another commonly used normalization. The values of attribute A are normalized based on $mean$ of attribute A values and $standard\_deviation$ of the attribute A values. The function looks as follows:

$$new\_value_A = \frac{value_A - mean(A)}{standard\_deviation(A)} \tag{2.2}$$

This normalization is useful when there are outliers that would infer the linear normalization or when the minimum or maximum of the attribute values is unknown.

Data Balancing

The datasets that are used for classification tasks are constructed of samples that belong to two or more classes. At least one class of interest exists in such datasets. However, the class of interest usually contains fewer data samples than the other with normal data samples. Such imbalanced datasets can cause models to be less sensitive to the minority data class. Hence, techniques such as sampling or data augmentation are involved.

To handle the imbalanced datasets, the majority class can be reduced using sampling techniques, and many different approaches to data sampling, such as random sampling, exist. On the other hand, new artificially created data can be added to the minority class. Techniques to augment data samples by simple duplication or more specialized Synthetic Minority Over-sampling (SMOTE) can be used. The techniques of sampling and augmentation can be combined to improve overall accuracy [23].

2.3.2 Machine Learning Algorithms

Many machine learning algorithms exist and can be divided into multiple categories. Two main categories that can be useful in the work are supervised and unsupervised learning methods. All the algorithms work with data samples that are, in this case, represented by the flow attributes.

Supervised learning for classification works with datasets containing data samples with a pre-assigned label. The number of labels depends on the number of so-called classes that should be distinguished. The labels are assigned either manually or automatically. The supervised algorithm is used to build a classification model upon the labeled data samples. The process of building a classification model is divided into two phases: training and testing. The labeled data are split into two parts, one devoted to each of the phases. The model is built on the training part and then validated on the unused testing part. The process of training and validation of the model has various approaches, which are not further described in this work.

The other category, unsupervised learning (clustering), does not require training and testing phases. The algorithms split the data into multiple classes based on the similarity. The algorithms take input parameters, e.g. number of classes and the input data. The data divided into multiple clusters is an output. These algorithms can work with unknown data, and hence it should be difficult to find the best parameters, such as the number of clusters for different tasks [104].

Only a brief description of algorithms that were later utilized in this work is provided. The description of the algorithms is primarily based on the knowledge taken from [47], [103], and [92]. Only statistical (shallow-learning) algorithms are considered. The neural networks (deep-learning) were not considered and used in this work since other algorithms proved to be very effective.

Decision Tree

The *Decision Tree* classification algorithm is one of the effective algorithms that is used as a base classifier in more complicated algorithms such as ensemble methods. A decision tree is a flow-chart-like tree structure. Every non-leaf node represents a test on an attribute of the tested data tuple. All leaf nodes hold the class labels. The topmost node is the root node. The algorithm learns decision tree induction from class-labeled training tuples. The approaches often used are ID3, C4.5, and CART. The decision trees are built in a top-down recursive divide-and-conquer manner using these approaches.

During the tree build phase, the algorithm is applied to a training data set of data tuples. The most promising attribute is chosen for each decision using a heuristic procedure. The procedure employs attribute selection measures such as *information gain*, *gain ratio* or *gini index*. The chosen procedure influences the tree structure if the tree is strictly binary or not. There should be a branch for each discrete value; in this case, the tree is a binary tree. The tree should also be binary for discrete values. For continuous-valued attributes, the split points are determined to split values for classes on different ranges.

The generated decision trees have a problem with *overfitting* the data. When the trees are generated from training tuples, there are many branches generated that reflect anomalies and noise in training data. This problem is solved by tree pruning. The tree pruning reduces the number of branches of the tree. The pruned trees are smaller and less complex. It is usually faster and better in the classification of independent testing data. There are several methods of pruning including prepruning and postpruning.

Tree-Based Algorithms

The *Random Forest* is a tree-based algorithm and it is one of the *ensemble methods*. Those methods combine a series of k base classifiers to create an improved composite classification model. The result of the ensemble is a class prediction based on the resulting vote from the base classifiers. The ensembles may produce more accurate classification, mitigating errors produced by individual base classifiers. In the case of ensembles, the error has to be in more than half of the base classifiers' results. The Random Forest classifier consists of decision trees as its base classifiers. Several techniques for improving the accuracy of this ensemble classifier exist.

One of the methods is *bagging*. To create multiple decision trees, different training tuples are chosen. For each decision tree, a bootstrap D_i of training data tuples d of training data set D is created. Some of the d tuples could be missing, and some of the d tuples may occur more than once, in resulting D_i training set to train a base decision tree. The other technique is to pick a random different set of attributes for each base decision tree.

Another method used to improve the accuracy of the ensemble classifiers is called *boosting*. The boosting technique works with each training tuple and assigns it a weight. The base classifiers are iteratively trained, and the weight of each tuple is changed upon the ability of the trained classifier to correctly classify the training tuple. When a training tuple is misclassified, the weight is increased so that more attention is given to the tuple in the next training iteration. The final ensemble classifier is a collection of base classifiers, and the weight of a vote is a function of its accuracy. Examples of ensembles using the boosting method are Adaboost and others.

An ensemble method does not solve the overfitting problem of the base decision trees. Hence, hyperparameters such as the maximum allowed depth of the trees should be carefully tuned.

Naïve Bayes

The *Naïve Bayes* classifier is another example of a classification algorithm that is often used in network traffic classification tasks [99, 135]. Based on a probability, bayesian classifiers predict if a sample tuple belongs to a particular class. The Naïve Bayes classifier performance and accuracy are comparable to Decision Trees and some Neural Network classifiers [47].

The Bayesian classifiers are based on the general Bayesian theorem 2.3. Let assume that X is a data tuple, called „evidence". The H is then some hypothesis that X belongs to some class C. $P(H \mid X)$ is the *posterior probability*, a probability that data tuple X belongs to class C.

$$P(H \mid X) = \frac{P(X \mid H)\, P(H)}{P(X)} \tag{2.3}$$

$P(H)$ is *prior probability* and it is independent of X. $P(H)$ is based on less information than posterior probability $P(H \mid X)$. $P(X)$ is then the prior probability of X. The $P(X \mid H)$ is then the posterior probability of X conditioned on H. [47]

The Naïve Bayes, or Simple Bayesian classifier is based on certain assumptions that features of the data vector are conditionally independent. In the case of this work the the individual flow is represented by x. Then there is a set of k possible flow classes $C = (C_1, C_2, ..., C_j, ..., C_k)$. The conditional probability for C_j could be then described as follows:

$$P(C_j \mid x) = \frac{P(x \mid C_j)\, P(C_j)}{\sum_{j=1}^{n} P(x \mid C_j)\, P(C_j)}. \tag{2.4}$$

Where $P(C_j)$ is the prior probability of class C_j and $P(x \mid C_j)$ is the conditional probability of flow x that belongs to class C_j.

It can be assumed that each flow x is a collection of attributes of flow then x is a vector $(A_1, A_2, ..., A_n)^T$ [140]. The conditional probability then can be written as:

$$P(x \mid C_j) = \prod_{k=1}^{n} P(A_n \mid C_j). \tag{2.5}$$

The equation then can be written as follows:

$$P(C_j \mid A_1, A_2, ..., A_n) = \frac{\prod_{k=1}^{n} P(A_n \mid C_j)\, P(C_j)}{\prod_{k=1}^{n} \sum_{j=1}^{n} P(A_n \mid C_j)\, P(C_j)}. \tag{2.6}$$

Naïve Bayes assume attribute independence. The attributes are continuous-valued and thus are subject of Gaussian distribution with a mean μ and standard deviation σ, defined by

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{2.7}$$

$$P(A_k \mid C_j) = g(A_k, \mu_{C_j}, \sigma_{C_j}). \tag{2.8}$$

Where A_k is one of the attributes. The μ_{C_j} and σ_{C_j} should be computed from values of attributes A_k that belong to a given class C_j [47].

K-Nearest Neighbors

The *K-Nearest Neighbors* classifier is a supervised learning method that works on a principle where it searches a k nearest data points (neighbors) in n dimensional space for an unknown data point. This way the unknown data point is identified as belonging to a certain category by finding the k closest neighbors belonging to that category.

The Euclidean distance is used as a metric determining the closeness of the data points [47]. Given two data points in n dimensional space, $X_1 = (x_{11}, x_{12}, ..., x_{1n})$ and $X_2 = (x_{21}, x_{22}, ..., x_{2n})$ the euclidean distance is defined as:

$$distance(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}. \tag{2.9}$$

The algorithm is sensitive to a combination of small-number attributes and large-number attributes. The large-number attributes can outweigh the smaller ones. In this case, the scaling of the input data should be employed before passing into the algorithm and computing the distances between the data points.

K-Means

The *K-Means* algorithm falls into the category of unsupervised machine learning. It partitions a set of n data tuples into k clusters based on similarity. The k is specified as an input parameter. The measure of similarity is based on the *mean* value of the objects in a cluster.

The algorithm randomly chooses k objects that represent the center of a cluster. In each iteration, the most similar object is taken and added to the cluster. The shortest distant object from the cluster center is chosen. The new mean representing the center of the cluster is computed. The algorithm iterates until the criterion function converges. The square-error criterion is typically used and it is defined as:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2. \tag{2.10}$$

Assume that p represents the object in space, m_i is the mean of the cluster C_i. The E then represents the sum of the square error of objects in the whole data set.

The method is sensitive to noise because the points that are more distant can significantly change the mean of the cluster. Hence, an appropriate input data normalization can be used to improve the accuracy of the method.

2.3.3 Performance Measurement

Many statistical metrics that can be used to determine the performance of the classification algorithm exist. In this work, only two classification classes are considered, the DoH and non-DoH flows.

- **True positive** (TP) refers to a number of positive flows that were correctly labeled.
- **True negative** (TN) refers to a number of negative flows that were correctly labeled.
- **False positive** (FP) refers to a number of negative flows that were incorrectly labeled as positive.
- **False negative** (FN) refers to a number of positive flows that were incorrectly labeled as negative.
- **Positive** (P) refers to the total number of positive flows.
- **Negative** (N) refers to the total number of negative flows.

The provided terms represent keys in the *confusion matrix* 2.2. The matrix provides well-arranged schema for analyzing the recognition of the classifier for tuples of different classes.

Table 2.2: Confusion matrix for two classes. P' and N' denotes predicted positives and predicted negatives respectively.

		Predicted		
		C_1	C_2	Total
Actual	C_1	TP	FN	P
	C_2	FP	TN	N
	Total	P'	N'	$P + N$

The final performance is measured on the unseen test part of the dataset. Based on the information depicted in the confusion matrix several statistical indicators can be computed. The first and most often used is *accuracy*. The accuracy measure should be used in cases where the testing data in classification classes are equally split. Otherwise, the results can produce non-representative performance. It can be defined by the equation:

$$accuracy(M) = \frac{TP + TN}{P + N}. \tag{2.11}$$

Where M is the classifier. The other statistic is an *error rate* that is simply $1 - accuracy(M)$. The error rate can be determined as a percentage of the incorrectly classified data tuples. It can be also defined by the equation:

$$error\_rate(M) = \frac{FP + FN}{P + N}. \tag{2.12}$$

This equation works well also for cases where the data in classes are balanced. For the other cases, *sensitivity* and *specificity* can be used to better reflect the reality. The sensitivity represents the proportion of positive tuples that are correctly identified. The specificity is the proportion of negative data tuples that are correctly identified. Those measures can be computed by the following equations:

$$sensitivity(M) = \frac{TP}{P}, \tag{2.13}$$

$$specificity(M) = \frac{TN}{N}. \tag{2.14}$$

The other two widely used measures are *precision* and *recall*. The precision computes what portion of tuples labeled as positive are actually positive. The recall gives a portion of positive tuples labeled as positive. The measurements are formulated as:

$$precision(M) = \frac{TP}{TP + FP}, \tag{2.15}$$

$$recall(M) = \frac{TP}{TP + FN} = \frac{TP}{P}. \tag{2.16}$$

The precision and recall can be combined into a single measure. It is $F1$ measure, that is computed by the following equation:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}. \tag{2.17}$$

The $F1$ measure represented by the equation 2.17 is the harmonic mean of precision and recall. It better reflects the performance of the results when samples of input classes are imbalanced.

Performance Measurement in the Context of Network Flow Detection

This work focuses on DoH detection in regular HTTPS traffic. To measure the performance of the detector, the three main metrics are important and used to measure the accuracy of the detector.

In the domain of security, especially network monitoring, where the solution is based on an assessment of individual flows, it is important to provide a high-accuracy solution. The flowing network traffic can reach high throughputs, counting thousands or even higher numbers of flow records per second. Even a very accurate solution reaching a 0.99 F1 score with a very low error rate can produce an immense amount of false positives and false negatives when considering the time frame of hours reaching up to thousands missclassified flows. Such an amount of misclassified flows can make even a very accurate solution impractical since the network operators have to investigate them.

The false positives are important, especially in the case of DoH detection, where the user would be prevented from accessing legitimate service when active action of blockage is connected to the detection. Hence, in this case, precision is another important measure that should be pushed to the maximum.

The F1 measure is used as a primary metric in this work with the aim to reach a desirable maximum near one and then precision that should also be maximized. The false positives produced by the detector would be decreased as much as possible, ideally down to 0. The ideal state would also be not to generate false negatives. However, this metric is less important and already partly represented by the F1 score. Furthermore, the confusion matrix is also provided since it represents well the absolute numbers.

2.3.4 Model-Centric and Data-Centric Concepts

In machine learning, there was established two concepts of how to deal with machine learning problems. Those can be called model-centric and data-centric.

The model-centric concept has been the main focus in the last years. The researchers focused on developing and designing machine learning algorithms, tuning their performance to the maximum. The measures were taken as a prize to be won [118]. Many practices as seed fixing and cherry picking, to reach maximum performance, were employed [82]. The researchers concentrated more on the models, feature engineering, and model architectures while the data was treated as a static asset [22].

However, in recent years, the researchers and teams realized that more emphasis should be on the data [121] used to develop successful solutions. The data-centric concept puts more emphasis on data that are used in machine learning models. In practice, the concept covers many techniques not only for improving datasets but also the general solution; some of the techniques can be recognized: understanding the data, better data collection, filtration of unnecessary samples that can ruin the models, data augmentation and many others [64]. The concepts cover error analysis, identifying the data that leads to errors, data consistency, and benchmarking.

This work inclines more to the data-centric concept. Hence, it focuses more on the understanding of the DoH communication, bringing the thorough domain knowledge (provided in Chapter 4 and Chapter 5) to be able to create comprehensive and representative datasets to work with. Moreover, domain knowledge helps to mitigate the influence of unwanted data by filtering the samples leading to errors before passing them into the classifier, helping the proposed method to be less prone to overfitting and making the final solution more practical.

Chapter 3

Related Work

Since the DoH specification in RFC 8484 [51] was published in October 2018, researchers have studied privacy, security, measuring performance, and other protocol characteristics.

DoH, as well as other DNS over Encryption protocols, brings many benefits to user privacy. However, as new protocols are being proposed and gaining popularity, many new challenges come. The survey work by Yan et al. [137] conducted in 2020 pointed to some challenges, including possible decentralization of the DNS ecosystem overtaken by large public resolver providers and lost control over some currently used local security mechanisms. Later, the survey work by Lyu et al. [86] summarizes the existing literature in the years 2016 to 2021, focusing on DNS encryption and security, providing a view into DNS over Encryption protocols, their performance, privacy, and misuse by malware. They pose concerns about the risk of untrustable resolvers providing DNS over Encryption services and monopolizing the DNS ecosystem. They suggest that the transition to new protocols comes at the risk of additional attacks related to TCP and TLS. Moreover, misconfiguration on the client or server-side may make the privacy mechanisms less effective. Hynek et al. [62] in their survey published in 2022 studying existing works defined still unsolved open challenges in this area that include: effective DoH blocking or filtering, detection of legitimate or illegal use, detection of malicious use, and detection of system bypassing. All the identified areas focus on increasing security in the computer networks and are mainly related to DoH.

This thesis aims to design practically usable and reliable DoH classifier by following a data-centric concept focusing on comprehensive analysis of DoH, understanding its environment, and creating datasets. The presented DoH related works are thus organized into three main categories covered in the following sections. The first focuses on the *analysis and performance measurement* of DoH, touching also other DNS and DNS over Encryption alternatives. The second category covers DoH from the *user privacy* perspective. The last category focuses on the *security* of DoH, aiming primarily at detecting DoH in regular HTTPS traffic and eventually detecting DoH tunnels and other malicious misuse.

3.1 Analysis and Performance Measurements

The migration from DNS towards DoH impacts latency and web browsing activities in different host environments using various providers as identified in a series of works by Hounsel et al. [54, 55]. Their measurements reveal that the DoH outperforms DoT for longer queries. Traditional DNS has a better response time than DoH and DoT mainly because of the overhead introduced by encrypted underlying transport channels of DoH and DoT. Finally, Hounsel et al. [56] later made a large-scale measurement in the US, showing that users do not necessarily need to trade their performance for privacy. The results show that DoT for page loading outperformed traditional DNS even though the resolution latency was higher. They also show that there was high variation when using different DoH providers. The performance was also varying across multiple ISPs. Around the world, performance measurement by Chhabra et al. [24] showed that bandwidth is the most significant contributor to performance degradation. Moreover, the DoH performance analysis made in Africa by Mbewe et al. [91] showed that other aspects such as location, caching, and even features like the TLS version and its traffic overhead directly impact the DoH performance. Another study focusing on DNS over Encryption performance comparison was made by Kosek et al. [77]. They performed experiments evaluating the most recent DoQ, other DNS over Encryption alternatives, and traditional DNS assessing influence on the page load time. They show that DoQ is about 10% faster than DoH, achieving similar performance as traditional DNS over UDP. Recently proposed protocol Oblivious DNS over HTTPS (ODoH) defined in RFC 9230 [74] designed to work in cooperation with DoH serves as a proxy for DoH. The protocol aims to protect users from relying on big resolver providers capable of identifying the clients' behavior. Singanamalla et al. [120] implemented the protocol, and they performed a series of experiments assessing page load times in browsers comparing the DoH with ODoH. They show that there is only a slight tradeoff in performance between using those protocols. The delays can vary based on the chosen ODoH server.

Other studies compared the performance of local and publicly available DNS and DoH resolvers. Callejo et al. [19] conducted an over-the-world measurement of the impact of DoH in comparison to traditional DNS in terms of page loading times. They found an increase in median resolution delays in tens of milliseconds depending on public DoH resolver provider compared to local DNS resolvers. Another study by Affinito et al. [3] compared local DNS resolvers of Italian ISPs to public resolvers. They showed that local resolvers outperformed the public ones, which is aligned with the previously mentioned work. They further compared the performance of public DNS to the DoH of two public resolver providers and showed almost no difference in their performance. Nevertheless, they also studied the ability of the resolvers to block malicious and phishing domains. All challenged resolvers showed promising blocking accuracy, but some public resolvers slightly outperformed the local ones. Doan et al. [40] focused solely on measuring DNS services, emphasizing public centralized DNS resolvers using an extensive network of RIPE Atlas probes. They show that about 30% of probes use public resolvers while the rest use locals. The public resolvers provide lower latencies as a benefit in most regions. The authors show still prevalent usage of local DNS resolvers, but they raise concerns that the portion of DNS centralization might change with higher adoption of DoT or DoH.

DoH protocol does not have an impact only on the performance as it brings another overhead. The DoH resolvers should also be analyzed. Since the proposal, the researchers have been focusing on this task. Deccio et al. [35] conducted a discovery of DoT and DoH resolvers and found 9 DoH resolvers. They tested only the TCP TFO feature of those servers. Lu et al. [83] made a similar attempt. They found 17 DoH resolvers, provided performance measurements, and mapped some basic features. They aimed at DoT as well. Bottger et al. [14] took ten currently provided open DoH resolvers from a well-known DoH curl list [126], performed a basic functionality check, and measured their performance. Those studies were made recently after the RFC 8484 proposal when only a few DoH servers were deployed. A more recent study by García et al. [45] performed a large-scale internet scan searching for operating DoH resolvers in 2021. The work was later extended in 2022 [44]. They discovered 4354 DoH resolver IP addresses and tested TLS certificates. In the same year, Luo et al. [84] made a similar scan and found 5715 DoH resolvers. They examined certificates and DNSSEC checks. Moreover, they focused on DoT as well. Both works included endpoints obtained from the curl DoH resolver list [126]. Others also focus on alternative DNS encryption protocols, particularly DoT and DoQ. Kosek et al. [76] periodically scanned the Internet for DoQ resolvers in 2021 and 2022 for 27 weeks, revealing an increase of DoQ responding endpoints reaching 1217.

DoH is implemented in the applications and operating systems because the protocol standard does not define a discovery mechanism; the applications come with a predefined setting that the users can change. The users' behavior concerning the DoH settings was studied by Nisenoff et al. [108]. They conducted a small-scale survey of users, showing that uninformed users mostly stick to the default DNS security setting. When informed about the technology, users tend to change the default settings either to switch off the security or change a provider. However, the ability of users to choose their own DoH provider is limited in some world locations, as shown by Basso [11] in his work. He measured DoT and DoH resolver endpoint blockage using a voluntarily hosted OONI Probe network. He discovered that the popular Cloudflare's and Google's DNS over Encryption services are the most blocked. Otherwise, he found that 80% of services were always reachable. The blocking occurred on various levels, either during the DNS resolution of those services or when attempting to establish a TLS connection.

Despite the considerable amount of studies focusing on DoH performance comparison, the influence of two RFC 8484 defined HTTP methods on DoH was not well studied. Moreover, another area that was covered briefly is single query DoH. The single query or short flow DoH connections are especially important since they can be used by malware like a Flubot [117] to hide their malicious domain resolutions. Moreover, many DoH resolver implementations and possible configurations complement the limited number of currently existing client implementations. The resolvers are the critical point for maintaining privacy and security of the whole ecosystem. Together with clients, they create variable combinations influencing the DoH traffic shape. The study of the network traffic characteristics and traffic shape should also be covered.

3.2 Privacy

The studies focusing on user privacy were trying to prove that fingerprinting or DNS pairing in encrypted DNS is possible and achievable even with currently designed counter mechanisms. The works [18,61] study pairing queries with responses in encrypted DNS channel pointing to insufficient DNS padding used in encryption was conducted in 2020. Moreover, Hynek et al. [61] show that using HTTP/2 for DoH transmission makes the pairing unreliable and nearly impossible. Siby et al. [119] attempt to show that traffic analysis-based monitoring and censoring of DNS is possible even when DoH or DoT are used. They show that standard padding schemes are ineffective. The study of Trevisan et al. [127] confirms the results of the previous works and points out that the padding of the sizes is insufficient to protect users' privacy. They have been able to attempt further attacks identifying the visited websites even with encrypted SNI. Dahanayaka et al. [32] showed a feasible website fingerprinting approach and presented the results of their proposed methods that achieved high accuracy on close-set and open-set datasets. They further studied the model's performance degradation over several weeks. Zou et al. [141] used a different approach. They proposed a machine learning model based on n-shot learning capable of distinguishing user-visited websites from DoH traffic using only a few traces to train the model. They showed high accuracy not only in closed environment but also in open environment. Another approach was proposed by Muhlauser et al. [102]. They studied the possibility of identifying Android applications using DoH or DoT encrypted DNS traces. They achieved very high accuracy when EDNS padding was not used. When EDNS padding was employed, the performance dropped significantly but was still above 70%. Unlike previous works, Niakanlahiji et al. [106] proposed a novel client-side obfuscation approach that helps prevent users from traffic analysis attacks and privacy leakage. Using techniques such as compression-aware padding, fake query injection, and random delaying of queries, they claim they were able to decrease traffic analysis accuracy from 95% to 9%.

3.3 Detection and Security

The works covered in the security category mainly include DoH detection approaches. The detection approaches focus on identifying DoH protocol and detecting malicious threads such as DNS tunnels inside DoH. The first mention of the necessity of DoH detection was in 2020 by Bumglang et al. [16] in their survey about the impact of mass DoH deployment. Garcia et al. [44], in their large internet measurement, also studied the completeness of publicly available DoH blocklists. According to their conclusions, IP-based DoH blocklisting is inefficient due to the incompleteness of publicly available blocklists.

Several researchers attempted to identify DoH mostly employing machine learning methods. One of the first DoH detection approaches that used teletraffic engineering to recognize DoH was proposed by Vekshin et al. [129]. They studied the shape of the DoH traffic using flows extended for information about the first 30 individual packets — packet lengths, packet times, Transmission Control Protocol (TCP) flags, and direction. They extracted 18 discriminatory features from those extended flows that proved efficient in DoH recognition, and their detector achieved an accuracy of 99.6%. Moreover, they also published their own dataset [128]. DoH detection was also studied by MontazeriShatoori et al. [97]. They use extended flows with 28 traffic features along with machine learning to distinguish DoH from regular HTTPS traffic. They evaluated multiple machine learning approaches, and the best one achieved an F1 score of 0.993. Moreover, they published the *CIRA-CIC-DoHBrw-2020* [96] dataset used during their study, which became the de-facto standard dataset for DoH detection. In addition to regular DoH, they proposed second-layer detection that had successful results in malicious DoH detection. Banadaki [10] designs his method using the *CIRA-CIC-DoHBrw-2020* dataset, which achieves an accuracy of 100%. His approach utilizes machine learning algorithms with time-related features. However, he also uses IP addresses and ports in its feature vectors, which is criticized by Behnke et al. [12]. Behnke et al. then improved the Banadaki proposal by further exploring his feature vector. They removed overfitting features (IP addresses and ports), further reduced statistically insignificant features, and finally measured several trained models. Their improved approach achieved a high F1 of 0.998. Following studies that also used *CIRA-CIC-DoHBrw-2020* dataset made by Casanova et al. [21], Jha et al. [72], Zebin et al. [138] and Mitsuhashi et al. [94] also performed well in DoH detection tasks achieving over 99% accuracy or 0.99 of F1 score. Konopa et al. [75] trained a standard multilayer forward neural network to distinguish between DoH, regular HTTPS, and HTTP. It achieved only 94.4% accuracy—significantly lower compared to other proposals. The authors of previously mentioned works explored the effectiveness of different machine learning algorithms and techniques to build their methods. A recent approach by Nguyen et al. [105] proposed a Transformer Neural Network to detect DoH. They used custom datasets and CICFlowmeter to extract 29 features and achieved an F1 score of 0.99. All of the mentioned approaches used statistical features extracted from flows.

Nevertheless, methods focusing on packet-based approaches were also proposed. One such technique was published by Wu et al. [136]. They used an autoencoder-based method to detect DoH resolvers. Using their approach they were successful in identifying DoH resolvers. Another packet-based approach was proposed by Csikor et al. [31]. They created a DNS over HTTPS classifier that achieved over 97% accuracy. They created their own large dataset to train and test the model's performance. They used metadata extracted from single packets and directly classified them. Since they work with packet lengths, they also study which padding strategy is best to counter the model and decrease its performance. Nijeboer [107] provided a simple algorithm with predefined thresholds detecting DoH from just packet sizes. The algorithm is combined with JA3 fingerprinting of the DoH resolvers.

Many other researchers focused solely on detecting DoH tunnels and malicious use in DoH traffic. Li et al. [81] evaluated several machine and deep learning classifiers on the task of DoH tunnel detection. Their later study [80] proposed a federated learning approach for DoH tunnel detection. They claim that the federated learning approach gives an advantage to preserving privacy but sharing knowledge between networks. They continued using the same *CIRA-CIC-DoHBrw-2020* dataset but transformed flows into figures that were then used for detection. Alenezi et al. [6] tested several machine learning models on DoH tunneling detection tasks. Moure et al. [100] used statistical traffic analysis and then designed a classifier for the same detection task. Simpler methods based on thresholds were also proposed. Kwan et al. [78] explored simple techniques based on thresholds to distinguish between benign DoH and tunnels. They propose a DNS tunneling prototype *dnsst* to circumvent possible censorship. Another method based on thresholds was proposed by Steadman et al. [124]. The method was aimed at detecting DoH data exfiltration attacks. Additionally, they used an IP-based prefilter to separate DoH from other HTTPS traffic. A different approach dealing with data exfiltration in DoH was presented by Zhan et al. [139]. Their method combined techniques of TLS client fingerprinting (benign clients as browsers or proxies) with a machine learning-based flow detector. They used their own captured dataset from different parts of the world. Khodjeava et al. [73] conducted several experiments using several publicly available datasets and proposed a method for detecting malicious DNS tunnels, including tunnels in DoH. Their experiments covered raw PCAP processing using multiple tools extracting statistical flow features enriched by entropy. Qiu et al. [112] also used DoH and HTTPS separation using IP addresses of known DoH resolvers, and they proposed a detection method for DoH tunnels based on a dual-tier classifier. In the first layer, they employ autoencoders to distinguish the benign and malicious flows. In the second layer, they identify a tunneling tool that generated the traffic. Du et al. [41] also employed autoencoders in their DoH tunnel detection approach. However, they work directly with packet metadata, where several packets are grouped into flows as input into the Bi-LSTM autoencoder. They claim that their approach is lightweight and capable of high processing speeds. Another technique using an autoencoder-based method working directly with packet metadata sequences for detecting DoH tunnels was presented by Ding et al. [38]. They treated DoH tunnels as an anomaly. They used a mix of publicly available datasets, and their specialty was using their bidirectional GRU-based network to automatically learn feature representations. All of the aforementioned methods for DoH tunnel or exfiltration detection achieved high F1 scores, reaching or crossing the line of 0.99. The majority of researchers were working with the *CIRA-CIC-DoHBrw-2020* dataset, sometimes combined with other publicly available datasets, or they created their custom dataset. All of the approaches worked with flow representation and statistical features extracted from flows, except for a few working directly with packet metadata.

Another work that can be seen as complementary to DoH detection is a study presented by Huang et al. [58]. They did experiments showing downgrade attacks that made browsers fall back from encrypted DoH to unencrypted DNS and made them successful. They show the possibility of downgrade attacks by blocking the resolution of DoH in most browsers. They argue a long recovery to DoH and recommend revising the implementation and DoH protocol. Since the DoH does not provide self-discovery mechanisms, the implementations use internally mapped IP addresses to prevent initial clean DNS resolution or require DNS resolution before using DoH. The successful downgrade attacks show the possibility of blocking detected DoH connections, preventing the implementations from recovery, and falling back to DNS, which can be inspected by current security solutions in enterprise networks.

Even though all mentioned studies focusing on DoH and DoH tunnel detections claimed very high accuracy, significant limitations still prevent their mass deployment. Either they are packet-based or require specialized data sources — flows extended for information about individual packets as in the case of Vekshin [129] or non-standard time-related features as created by DoHLyzer tool and provided within *CIRA-CIC-DoHBrw-2020* dataset or by *CICFlowmeter*, which are then used by majority existing studies [10, 12, 21, 94, 97, 105, 136]. These approaches require features such as median or mode of packet lengths, which cannot be computed on the running sequence. It means that the monitoring apparatus needs to hold the information about all packets in the memory. The computational and memory complexity prevents its deployment on monitoring infrastructures where hardware resources are limited—router or high-speed monitoring probes. The only approach that relies on standard telemetry data was proposed by Konopa et al. [75], who trained a standard multilayer forward neural network to distinguish between DoH, regular HTTPS, and HTTP. Nevertheless, it achieved only 94.4% accuracy— significantly lower than other proposals.

Chapter 4

Well-Known DoH Resolvers Analysis

This work focus at the problem of DoH detection in a more data-centric way. Hence, understanding the communication participants creating the data is very beneficial. The study of well-known DoH resolvers creates the first building block on which further analysis stands. It aims to show different characteristics that may influence DoH communication. Covering possible variability of server-side communication helps determine the different aspects that should be considered for creating more comprehensive datasets for DoH detection and analysis of the DoH traffic shape. Moreover, several security and privacy tests covered in this chapter show that some publicly available DoH resolvers represent a potential security risk, further amplifying the need for their regulation from the perspective of network operators.

Since the DoH protocol proposal in 2018, it has been implemented in browsers, proxies, and operating systems. With traditional DNS, the host has been automatically configured to a preferred resolver as provided by the DHCP option, or users might choose a custom from a variety of publicly available DNS resolvers [20]. In the case of the DoH, the automatic setting of the local DNS resolver is not an option, even if some implementations check whether the automatically set resolver supports DoH. The applications or systems come with a few predefined public DoH resolvers they use by default. However, users have the option to choose custom. Users who want to choose a custom server for DoH resolution depend on publicly available lists of DoH resolvers. Such lists are manually maintained and contain resolvers known to the community; hence, they are named as well-known. They can be taken as a representative sample.

There are currently several widely deployed open-source client implementations [126]. On the other hand, the situation of deployed DoH resolvers is less transparent due to private implementations and specific settings managed by different providers that affect the performance and security of the entire ecosystem [44, 84].

This chapter is based on article [69] published by the author of this thesis. The chapter studies the well-known DoH resolvers from the most comprehensive publicly available list maintained by the curl community [126], uncovering their internal characteristics by long-term monitoring and challenging the servers with various additional measurements. Over 500 unique DoH resolvers were tested over the period of 2 years, from September 2020 to December 2022. The presented results extend the previous studies by the number of resolvers, studying not covered aspects, such as long-term IP discovery important for block list creation, resolvers feature progression, HTTP headers, and EDNS padding influencing security, privacy, effectiveness, and traffic shape of DoH communication.

4.1 Methodology

The well-known resolvers exploration and analysis are done in two phases. The first phase focuses on long-term resolvers monitoring, covering only basic properties (Section 4.1.1). The second phase further challenges the servers to obtain a detailed analysis of TLS certificates, other DNS protocol support, HTTP headers, and EDNS padding (Section 4.1.2). The output from both phases represents novel comprehensive long-term data containing many DoH resolvers suitable for further analysis (Section 4.2).

4.1.1 Long-Term DoH Resolvers Monitoring

For more than two years, from September 2020 to December 2022 (27 months in total)[1], well-known DoH resolvers from the list maintained by the curl community [126] were monitored. The curl community list was chosen as the most comprehensive among existing actively maintained DoH resolver lists. Other maintained lists, such as the AdGuard list [2] contained only 76 resolvers, and the DNSCrypt list [39] contained 127 resolvers. Other lists contain only a subset of resolvers provided in the monitored list. In addition, the curl list is cited in many related works [14, 44, 45, 83, 84] and hence potentially receiving higher attention even by users.

The monitoring consisted of actively probing the resolvers and testing the availability of the features presented in Table 4.1. In addition to only probing for IP support, the discovered IP addresses were also collected.

Table 4.1: Summary of tested categories with specific versions and methods.

Probing Categories	Specifics
Supported TLS versions	1.0, 1.1, 1.2, 1.3
Supported HTTP methods	GET, POST
Available via	IPv4, IPv6
Supported HTTP version	HTTP/2
Provided certificates	Let's Encrypt Certificate

The monitoring occurred from the device attached to the Brno University of Technology network. The resolver testing was powered by a Python script utilizing dnslib[2] and running OpenSSL[3] library to test TLS. The domain names were resolved using Google DNS service over standard UDP. The checks were run once per hour. Extracted data were fed into the database. All records were extended for the timestamp of their insertion so the results could be tracked in time.

The collected data were cleaned before analysis. Inactive endpoints remained in the monitoring in case of their renewed activity. Hence, inactive endpoint records were removed from the data, and only records that satisfied two conditions remained: endpoint correctly responded to both DoH POST and GET queries (RFC 8484 defines that those two methods must be implemented), and they supported at least one TLS version (RFC 8484 mentions in protocol design requirements that the protocol must use HTTPS secure transport) [51].

[1]Note that in January and February 2021, the server running the tool was under maintenance; thus, values from this period are missing.

[2]https://github.com/paulc/dnslib

[3]https://github.com/openssl/openssl

4.1.2 DoH Resolvers Properties Testing

The resolvers identified as active in the last month of the monitoring period were further examined. The resolvers were challenged to uncover the following aspects:

1. TLS certificates inspection;
2. support of other DNS and DNS over Encryption protocols;
3. HTTP header sizes, variety, and minimalism in responses;
4. EDNS padding capability and strategies.

The active probing occurred on a separate measurement point in the same university network. All the measurements were done within one day.

The measurements are further divided into two categories. The first category gathers information about TLS certificates and other DNS protocol support. These properties are affected by the resolver domain and are shared by all resolvers on that domain. The second category focuses on resolver-specific features and utilizes regular DoH queries.

Tests Towards Resolver Domain

The measurements were done for a set of DoH resolver domain names. Although domains can be resolved to pools of IP addresses, the reason might be for load-balancing purposes, and all these servers run the same deployment. The aim is to get TLS certificates and determine whether the DoH resolvers behind the domain are open to other DNS resolution protocols.

The TLS certificate inspection was done using *testssl.sh* tool[4]. The tool can test the domains in parallel based on the list of domains. The testing tool was restricted to testing only TLS certificates. The tool could not connect to 18 cases, even after several retrials; hence that data is missing.

Secondly, tests whether the domains also host other DNS resolution options than only DoH was made. Standard DNS over UDP (DoUDP, UDP, port 53), DNS over TCP (DoTCP, TCP, port 53) were queried together with its encrypted alternatives DNS over TLS (DoT, TCP, port 853) and most recently proposed DNS over QUIC (DoQ, UDP, port 853). The domains were queried using dig[5], which was set for three retries on failure, and q[6], a publicly available command line tool, also used to query 3 times on failure.

Tests Towards Resolver

The data transmitted over the encrypted channel consisting of HTTP headers and the DNS message (particularly the EDNS padding feature) were of interest. Hence, a set of DoH resolvers was queried.

The resolvers were queried using a Python script utilizing the *dnslib* library. The servers were challenged with correct DoH queries using the POST method, sending only minimal HTTP headers required *content-type* and *accept* both with *application/dns-message* value. Resolvers were tested using both HTTP/1.1 and HTTP/2 versions since many servers can operate both.

The *HTTP Header* test case collects headers from each correct query response using HTTP/1.1 and HTTP/2 protocols. For HTTP/2, the test was repeated to compare the possible differences between header values sent in each response over the same connection.

[4]https://testssl.sh
[5]https://linux.die.net/man/1/dig
[6]https://github.com/natesales/q

In the *EDNS* test case, experiments with queries of different EDNS padding sizes were done. According to RFC 7830, the responder must pad the DNS message if the sender uses padding; otherwise, it violates the standard [89]. Several strategies for EDNS padding size responses can be expected in replies, some recommended in RFC 8467 [90]. Several padding cases were tested: no padding, padding in multiples of 128 and 468, and padding with a power of two from 0 to 2048.

4.1.3 Limitations

Despite the careful design of the methodology, there are some limitations. All experiments are run from a single client location, which can provide skewed data if geolocation DNS serving, firewall filtering, or other measures are applied to the client's address. Those mechanisms can impact the results so that not all IPs would be discovered, and some resolvers might be falsely marked as inactive or not working.

Potential blocking of the client address was mitigated by performing slow long-term querying of the servers done once per hour, which resembles a robot scraping activity pattern or lifetime checks and hence be considered harmless and generally allowed to operate. Still, DoH server monitoring checks can miss some DoH resolvers. The final active testing could create more queries quickly, even though a limit on the number of requests was set.

Another limitation is that the last seen timestamp was not attached to domain IP address mapping; hence, only IPs' first discovery times per endpoint can be presented.

4.2 Results

This section provides insight into the results of the long-term (over 27 months) periodic measurement of well-known DoH resolvers. In addition, it presents the results of other extended resolver challenges carried out at the end of the monitoring period.

4.2.1 Monitoring Insights

The monitoring relies on the DoH resolvers list manually maintained by the curl community [126]. Over the monitoring period, the list was continually extended. New resolvers were irregularly added as the community discovered and reported them. Figure 4.1 depicts the cumulative increase of unique resolvers as they were observed and added to monitoring. Since the list was maintained manually, some endpoints that appeared on the list were not responding correctly or were not active anymore. Sometimes, they remained on the list with an inactive flag to notify users that they were not active. A slowly increasing gap between active and inactive servers can be observed.

Over the whole period, 480 endpoints were seen and measured, from which 431 correctly responded at least part of one month. More than half of the resolvers, from mostly smaller and less known resolver providers, were added in November 2022. At the end of the measurement, 378 endpoints responded correctly and were considered for further investigation.

Together with correct responses, several other fundamental features were tested. Figure 4.2 presents feature support percentage among the active resolvers each month.

TLS 1.2 has almost 100% adoption, and the support of TLS 1.3 in deployments is also high, about 90%. Obsolete and deprecated TLS 1.0 and 1.1 are supported by less than 30%, slowly degraded to 11% minimum in July 2022, and slightly increased with new servers added to the list. TLS 1.0 and TLS 1.1 are not solely deployed but accompany newer TLS versions.

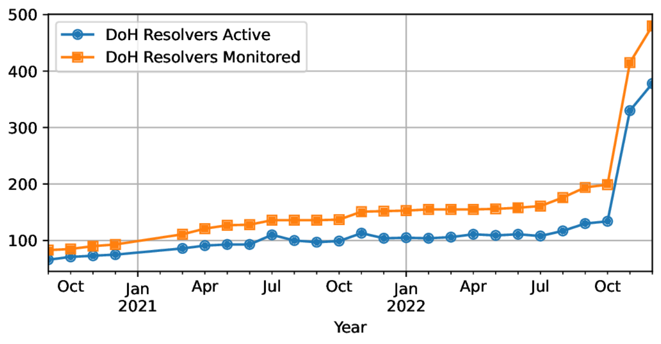


Figure 4.1: Cumulative increase of resolvers over the whole monitoring period.

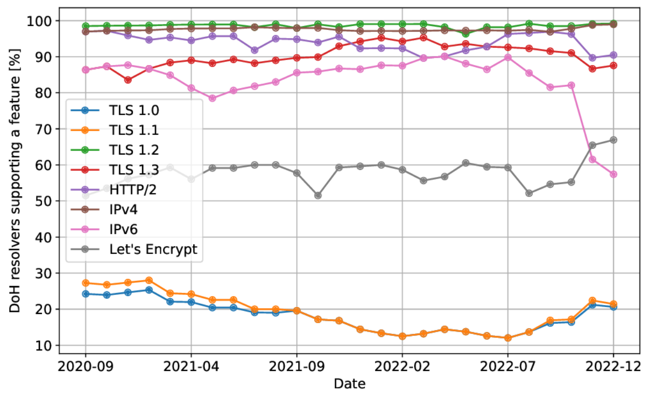


Figure 4.2: DoH resolvers features ratio captured over time.

Moreover, the *Let's Encrypt* certificates deployment was the most prevalent as they represent the most used provider of automated SSL certificate deployment on servers [1]. It seems that around 60% of resolvers are using such certificates.

Some resolvers do not support HTTP/2 despite being the preferred DoH [51] transmission option, mainly due to its performance. While only 2% of endpoints missed HTTP/2 support initially, it raised up to 10% of deployments over the whole monitoring period. Almost 100% of servers are available via IPv4, while IPv6 was configured on 80% to 90% of endpoints, with a significant drop at the end of measurement to under 60% as new resolvers were added to the list. Some resolvers were provided solely via IPv6.

4.2.2 IP Discovery

The results of periodic DNS queries are shown in Figure 4.3. It captures the progress of discovering new IP addresses for each resolver and discovering new resolvers. Each spike value above zero represents the number of addresses or resolvers found daily. Days without discoveries were filled with zero values. The inconsistency of the initial deployment causes a gap after the first discoveries of IPs and endpoints.

It can be observed that new IP addresses are not solely aligned with new resolver findings but mostly continue for more days. New IP addresses still pop up even after several days. Some resolvers are available on several IP addresses, and DNS load balancing with Round-robin or different mechanisms is employed. Another case of this phenomenon can occur when the resolver frequently changes IPs, e.g., when deployed on dynamic cloud infrastructure. All endpoint IP addresses are

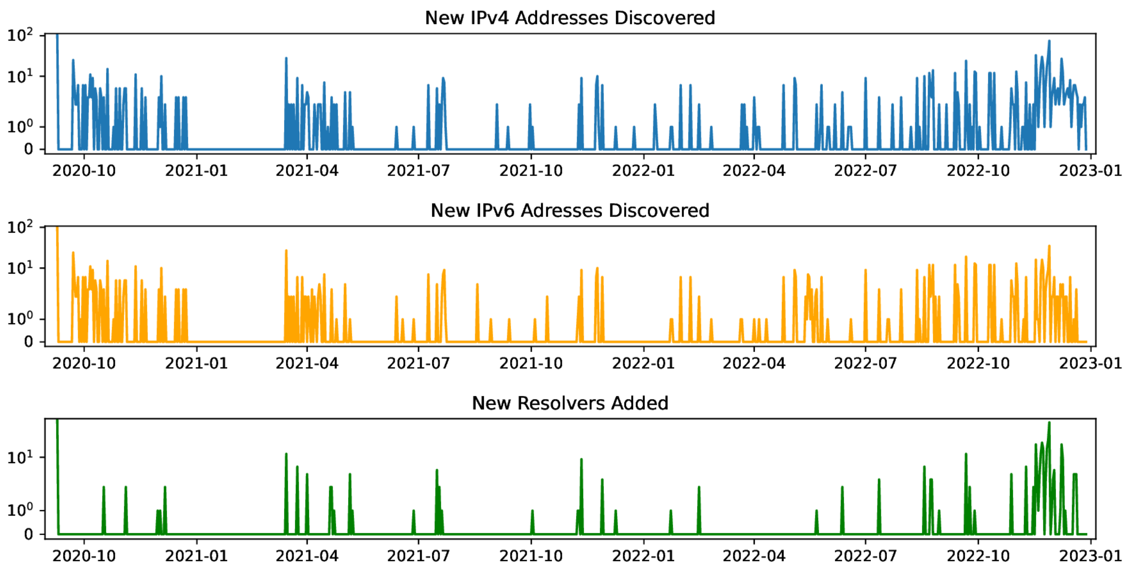


Figure 4.3: Ammount of DoH resolvers added together with IP address discovery over time, displayed using logarithmic scale.

mostly discovered in a few days with no later findings. In addition, some isolated extremes can be observed. Those mostly occur when some changes happen in stabilized deployment.

In total, the graph contains 1994 IP addresses (862 IPv4 and 1132 IPv6) and 431 resolvers. IP addresses are not unique since per endpoint collection was used; hence, some share the same IP. The unique IP addresses are 1405 (834 IPv4 and 571 IPv6).

4.2.3 End of Monitoring Summary

At the end of the measurement period, the data from the last month (December 2022) was analyzed. Summary information covering several aspects of the most current state of the well-known resolvers is provided in Table 4.2.

All measures are extracted from 378 resolvers that satisfied the DoH correctness condition. The total amount of resolvers resides on 355 unique resolver domains. The unique domains are divided into multiple subdomain levels, splitting the 281 unique second-level domains (SLD). According to the curl community list, the domains are managed by 276 providers, with 42 providers managing more than one endpoint.

DoH resolvers are, in more than half of cases, available on both IP versions. Only four endpoints provide access solely via IPv6, and 161 only via IPv4. Moreover, 40 resolver domains are deployed on shared infrastructure, sharing IP addresses with at least one other domain. Those domains can be divided into 17 groups sharing the same IPs, which makes 2.3 domains on average served on the same IPs within the 40.

Additionally, 11 domains provide DoH resolution services on multiple URL paths simultaneously, which differs from the official RFC 8484 definition. Despite domains sharing multiple URL paths within the same domain, other resolvers also choose non-officially defined paths to serve DoH.

Table 4.3 provides insight into the URI paths used. Most of the resolvers follow the RFC 8484 with official *dns-query*, and the second most often is / path. The other category consists of infrequent paths such as *uncensored*, *adblock*, *adultfilter*, *ads*, and *family*, mostly describing their purpose. Some not intuitive ones include *p0* and *x-oisd*.

Table 4.2: Summary of data analysis results from last monitoring month.

	#	%
Total unique endpoints active	378	100
Both IPvs endpoints	213	56.3
Only IPv4 endpoints	161	42.6
Only IPv6 endpoints	4	1.1
Total unique domains	355	100
Unique SLD domains	281	79.2
Unique domains sharing IPs	40	11.3
Unique domains with multiple paths	11	3.1
Total providers	276	100
Providers with more than one endpoint	42	15.2

Table 4.3: Portions of RFC 8484 defined path and other paths found among active resolvers.

URL Path	#	%
dns-query	328	86.77
/	22	5.8
other	28	7.4

4.2.4 TLS Certificates

DNS over HTTPS is closely bound to HTTPS and hence TLS to provide confidentiality to the DNS protocol. As such, the well-known resolvers certificate checks were done to asses validity, whether security standards are met, and the identity of the issuers. Data from 18 resolvers were incomplete since they did not respond, leaving us with certificate information from 337 domains.

The testing identified certification authorities (CA) issuing the TLS certificates for DoH resolvers. Long-term monitoring already revealed that the certificates Let's Encrypt CA issued cover more than 60% of all endpoints. The second most represented CA issues free certificates using an automated process and covers another 10%. Cloudflare's 3.3% of free certificates for their hosted services create almost 80% of free automated, well-known DoH resolvers' TLS deployments. The rest is covered mainly by paid CA with a more complicated certificate issuing process, as presented in Table 4.4.

Table 4.4: Portions of TLS certificate CA names among certificates acquired during testing.

CA Name	#	%	**CA Name**	#	%
1) Let's Encrypt	221	65	6) Cloudflare	11	3.3
2) ZeroSSL	36	10.7	7) Google	9	2.7
3) DigiCert	20	5.9	8) GlobalSign	7	2.1
4) other	15	4.5	9) Buypass	5	1.5
5) Sectigo	13	3.9			

The 6 endpoints had issues with their certificates. Five endpoints suffered from expired certificates; one had an incomplete certificate chain. The issues were reported to administrators of the problematic DoH services. No other certificate-related problems were found.

4.2.5 Support of Other DNS-Related Protocols

Results of the DoUDP, DoTCP, with its encrypted complements DoT, DoQ, queries were analyzed, and the results are presented in Table 4.5. Note that all tested endpoints already satisfied the DoH deployment condition.

Table 4.5: Amounts of DoH resolver domain hosting services responding to other DNS protocols.

Protocol	**Only**	**Both**	**At least one**
DoUDP	151	146	169
DoTCP	164		
DoQ	29	28	251
DoT	251		
All together	13		
At least one	276		

Table 4.5 shows that over three-quarters of the resolvers' domains host at least one other service to resolve DNS. Most of the resolvers' domain services also run longer existing DoT. Surprisingly, 29 resolver domains also already host only recently standardized DoQ. At the same time, less than half of the resolvers' domain services provide unencrypted versions of the DNS. Only 13 support all tested DNS resolution options, and 276 support at least one other, leaving 79 resolvers' domains hosting DoH solely.

4.2.6 HTTP Headers

The HTTP Headers are an integral part of DoH, without which the DoH can not operate. The results of the headers returned in the resolver's query response and their content were analyzed. HTTP/1.1 headers were analyzed first. Its content showed almost the same results and amount with HTTP/2 headers, except for a possibly different impact on the number of headers exchanged during the connection when the HPACK compression mechanism is involved.

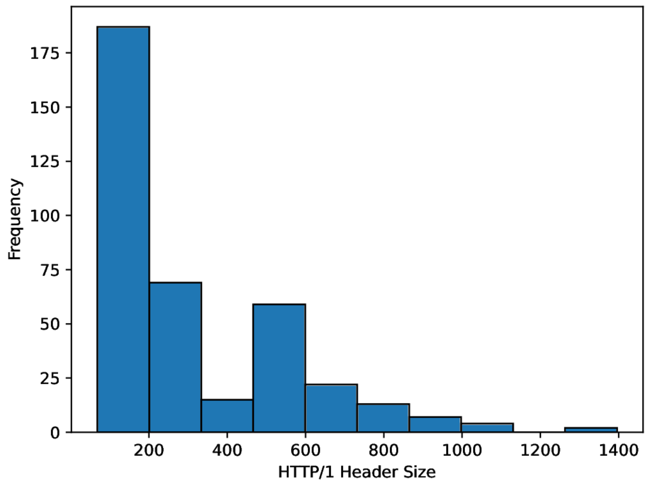


Figure 4.4: HTTP/1.1 response header sizes of all endpoints.

HTTP/1.1 header sizes can be easily computed from the obtained DoH responses. The histogram presented in Figure 4.4 depicts header sizes obtained from all endpoints. At least half of the resolvers' configurations keep the response headers minimal, reaching up to 200 bytes. However, some configurations stuff the response with headers reaching up to 1000 bytes.

While all current clients are sending only minimal HTTP headers necessary for proper DoH functionality, limiting to *content-type*, *accept*, and *content-length*, the DoH resolvers are responding with a huge variety of headers. Detailed insight into frequently used HTTP headers is provided in Figure 4.5. The graph represents the number of each header name provided in all responses. The headers with less than ten occurrences were discarded from the view as most were individual occurrences.

Based on the analysis of HTTPS headers, the following header categories in the responses were identified:

- **DoH Related** - Headers belonging to this category are the HTTP, or DoH recommended or mandatory covering *content-type*, *content-length*, *date*, and those related to caching and potentially security.
- **Unrelated** - Some endpoints are sending the headers, in our opinion, not related to DoH, such as *content-security-policy*, especially when the endpoints are serving only DoH.
- **Potentially Harmful** - Headers such as *expect-ct*, *x-powered-by*, and *server*, quite often seen in the replies, are recommended to be removed by OWASP [109].
- **Privacy** - Even though *report-to* and *nel* headers serve for monitoring purposes [17], they use the clients to send network-related information actively based on the server's requested policy without the user's knowledge. The reporting mechanism is incorporated directly into the browser reporting information. The information is mostly sent to third-party monitoring appliances, potentially causing privacy leakage [133, 134].

The header values revealed some useful information. The *report-to* headers were all routed to Cloudflare's monitoring endpoints accompanied by *nel* with default policy settings instructing clients to report all network failures and no successes for the domain they were using. Provided *server* and *x-powered-by* headers revealed software that powers the DoH endpoints. The *server* header uncovered 53 AdguardDNS, 52 h20/dnsdist[7], 27 cloudflare, and 6 DNS-over-HTTPS[8] in the specific version used, the rest values contained either apache, nginx, or changed values as recommended by OWASP [109]. The *x-powered-by* revealed additional 60 DNS-over-HTTPS with a version. None of the resolvers send any cookie-related headers.

When HTTP/2 is used, the amount of transmitted headers is reduced thanks to the HPACK compression mechanism [110]. The transmitted header sizes will depend on the HPACK implementation and settings. However, when a high number of headers is transmitted, the effectivity may drop with some configurations, causing unnecessary header retransmission. In addition, some headers change values over requests and would be included in each exchange with the required ones.

We analyzed two exchanges within one connection over HTTP/2 towards all HTTP/2 capable DoH resolvers and identified several headers often transmitted repeatedly containing different values. Some of the headers are *date*, *cache-control*, *expires*, and others. Those can influence the traffic shape of DoH clients communicating with different resolvers.

[7]https://dnsdist.org/guides/dns-over-https.html
[8]https://github.com/m13253/dns-over-https

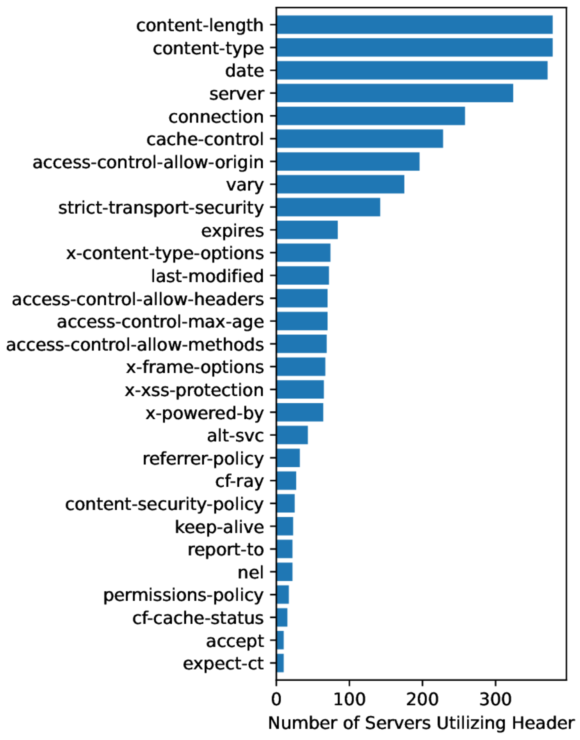


Figure 4.5: Detailed view into the frequency of specific HTTP header occurring in DoH resolvers replies.

4.2.7 EDNS Padding

EDNS padding testing results show that only 66 resolvers support this feature. The rest answer the queries but do not perform padding in the answers.

Table 4.6: EDNS padding strategies used by DoH resolvers together with their amounts revealed during testing.

Strategy	**#**	**%**
Not replying with padding	312	82.5
Replying with padding	66	17.5
Padding message to multiples of 468	35	53
Padding message to multiples of 64 plus 4	21	31.8
Reply half query padding size	6	9.1
Random padding size	4	6.1

The resolvers providing DoH response EDNS padding used four different strategies as summarized in Table 4.6. Most resolvers were stuffing the DNS responses to have the size of 468 bytes multiples. The second common strategy was padding the size to multiples of 64 plus an additional 4 bytes. The last two strategies were to reply with random padding or use padding of half the size of the requests padding size. Most of the servers could respond to messages padded up to 1024 size; otherwise responded without padding or with error.

4.3 Discussion

The results show that the monitored well-known resolvers keep a stable portion of various feature support among the servers. Only the end of the monitoring period showed slight drops in IPv6 and HTTP/2 feature coverages and increases in TLS 1.0, TLS 1.1, and *Let's encrypt* certificates support as new resolvers were added to the list.

Resolvers using deprecated TLS 1.0 and TLS 1.1 versions may be subject to downgrade attacks since the older versions of TLS represent security risks as they rely on weak cryptography algorithms that can be utilized in man-in-the-middle attacks. Their usage may lower the security of the DoH. It can be expected that the deprecated versions are supported intentionally for possible backward compatibility with older versions of the OpenSSL library on clients.

The long-term resolvers IP discovery revealed that it might take a few days to discover all endpoint's IP addresses due to DNS load-balancing and other dynamic DNS mechanisms. This fact may be considered when creating DoH resolvers label lists on which the existence of some proposed DoH works [38, 124] rely. The data also support the premise of García et al. [44] that the environment of DoH resolvers is volatile and constantly changing as new resolvers are added, and others disappear.

Almost 80% of the servers use CAs that are issuing certificates for free, even using automated deployment. The portion of resolvers using free CA issuers is also similar to the results of García et al. [44] although they examined much more resolver IP addresses from the whole internet scan. The huge employment of free automated certificate deployment makes the trustworthiness of such services questionable since many potentially malicious actors can hide their deployments under the certificates of such loosely verifying CAs. Only six endpoints had an issue with the certificates. The operators of the problematic resolvers were notified about the issues.

The results are aligned with a recently published study of DoH resolvers internet scan performed by García et al. [44] and also with other works [16, 84] statements arguing the necessity for DoH detection and resolvers discovery. Not all resolvers are necessarily secure and represent a risk to users' privacy. The users and companies should choose carefully which DoH resolvers can be permitted. This is also aligned with the DoH detection task that can help enforce security policy and hence increase security.

The well-known resolvers are taken in this work as a representative sample of publicly available DoH resolvers present on the Internet available to the users. The results show that not all endpoints comply with RFC 8484 official URL path *dns-query*. In addition to the official URL, the root path would also be considered in DoH resolver discovery as it represents 5.8% of cases. Moreover, the DoH may be served from multiple paths within one domain, and multiple domains can be served from the same IP address. Which may complicate the verification of the endpoints whether they are providing DoH services. The verification of the RFC-compliant URL path might be limiting, and some resolvers may be missed. More common paths may be tested, at least including the root path that is the second most common.

The analysis shows quite a huge variety of HTTP headers that the resolvers are sending the small headers sent by the clients. More than half of the tested resolvers are sending unnecessary high headers that should have an impact on the traffic shape of the communication. Another unveiled characteristic shows that only 17.5% of DoH resolvers were capable of responding with EDNS padding. Resolvers used four different strategies of EDNS padding. Related works [24, 91] presented that the bandwidth significantly influences the DoH performance and also traffic shape. The resolvers responding with different amounts of headers and implementing different EDNS padding strategies should be used when creating a comprehensive DoH dataset. The variability coverage in the data contributes to proper traffic analysis and the later creation of the detection method for this work.

Chapter 5

DoH Measurements and Traffic Analysis

The previous chapter provided a thorough analysis of the server side of the DoH ecosystem required for understanding the target domain. This chapter, along with the followed data-centric approach used in this thesis, complements the knowledge by detailed analysis of DoH communication patterns and puts more emphasis on the client side of the DoH environment. The work also focuses on creating a more practical detection method with limited information extracted by existing monitoring infrastructure. Hence, the understanding of the DoH communication shape and difference between the rest of the HTTPS traffic should help to use the limited data source more effectively. Presented results covered in this chapter were also published in article [68] published by the author.

Many works were published on performance comparison of DoH to other protocols, especially with former unencrypted DNS protocol. The works focused on various aspects of studying performance of various protocols [54–56, 77, 120], different aspects and locations [24, 91] and comparing local resolvers with public ones [3, 19]. The works did not study the influence of HTTP methods POST and GET on DoH resolution.

DoH is designed to be used with HTTP/2 to use the protocol's advantages that introduced communication multiplexed across multiple streams and other effective mechanisms. DoH using HTTP/2 is meant to utilize long-living connections, pushing all necessary queries over a single encrypted channel, amortizing the initial overhead introduced by TLS handshake and HTTP/2 preface. However, the technology does not prohibit the clients from using single query connections. Those can be used by applications such as malware to hide their activities.

This chapter aims to cover the problem of single DoH queries and the influence of DoH using two defined HTTP methods (POST and GET), and DoH traffic analysis. Moreover, since the previous Chapter 4 brought the analysis of DoH resolvers, this chapter builds on the knowledge and assesses multiple resolvers taken from well-known resolvers with various characteristics during the traffic analysis. Compared to the previous chapter, this chapter focuses more on the client side of the DoH ecosystem. The chapter brings a complementary knowledge of DoH.

Four different analysis of DoH that was not thoroughly covered by existing works are presented here. The first analysis aims to analyze single DoH queries as those can be used by smaller applications and malware to resolve domains. The second shows the behavior of browser implementations covering the most advanced client implementations. The third analysis focuses on the impact of DoH resolution using two different HTTP methods, POST and GET, which were not studied before, showing whether there is a significant difference. The last analysis builds on all prior knowledge and studies the traffic characteristics of DoH.

5.1 Single DoH Query Analysis

The first experiments aim at measuring single DoH requests. The experiments estimate the expected overhead of DoH compared to the traditional DNS transaction. Moreover, it shows the potential length and other characteristics of the connections. Results can be used to set the baseline for the minimal length of connections where actual DNS data can be transmitted or identification of unsuccessful DoH requests generated by browsers.

The RFC 8484 [51] defines that the DoH supports two HTTP methods to be used for querying: GET or POST. In addition to those two methods, an experimental DoH with JSON encoding is supported by big DNS providers such as Cloudflare, Google, etc. Thus, JSON-encoded DoH is considered as another variant in experiments since it may also be used in single query connections.

5.1.1 Methodology

The domain names used in queries stem from the top one million domains from the Cisco Umbrella dataset [26]. Only resolvable domains were included in the dataset, yielding approximately $800,000$ domain queries for each type of request method. All queries were done towards Cloudflare servers.

Only a few standalone tools can perform just a single query. Those tools are command-line tools such as *nslookup* (for DNS queries), Curl community-developed simple *curl-doh* (for DoH), and my simple command-line tool for querying DoH for both GET and POST commands. Using browsers and proxies is unsuitable for this task since the number of queries in a single flow can not be controlled.

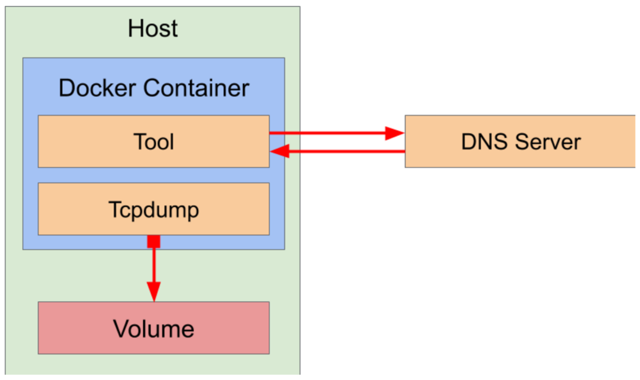
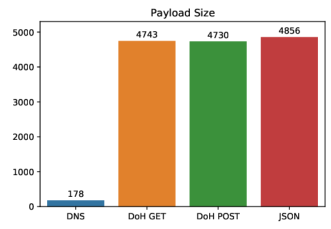


Figure 5.1: Container structure for data generation.

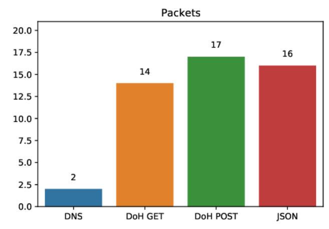
A Docker container that enables capturing network traffic on its interface (see Figure 5.1) was used to run the toolset. The hardware platform was Supermicro SuperTwin2 6026TT-TF server equipped with eight Intel (R) Xeon E5520 @ 2.26 GHz. The cluster consists of 4 nodes. The nodes are equipped with 48 GB RAM and 16 CPU cores. The cluster is connected via 1Gb/s links to the campus network. The tools were run simultaneously on all cluster nodes. The resulting data are published as part of the dataset [71].

5.1.2 Results

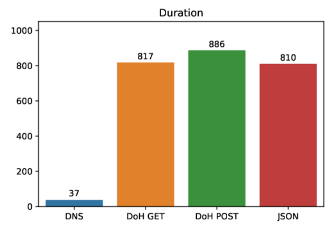
The captured packets were aggregated into flows. The following flow statistics were computed: payload size, number of packets, and overall flow duration. As flow duration follows the normal positively skewed distribution, medians were used for comparison rather than means. Figure 5.2 consists of graphs showing the differences of DoH using POST, GET, and JSON. Note that the DNS



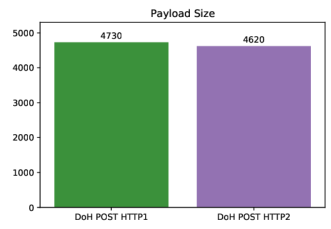
(a) Median payload size of flows comparison.



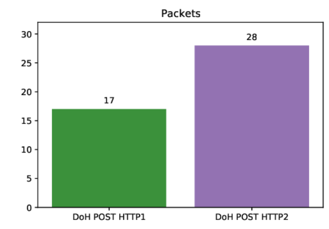
(b) Median packets in flows comparison.



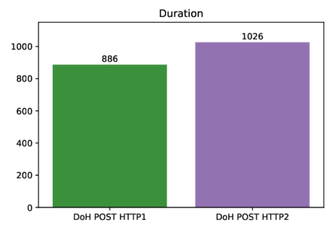
(c) Median duration of flows comparison.



(d) Median payload size of flows comparison.



(e) Median packets in flows comparison.



(f) Median duration of flows comparison.

Figure 5.2: Comparison of DNS and DoH options statistics.

in JSON is used here only as a reference since it would represent similar characteristics to small REST API calls. Additionally, the metrics are compared with the traditional DNS.

Figure 5.2a show the medians of the flow payload sizes for single pairs of DNS queries. To transmit a single DNS query and response of about 178 bytes using clear DNS, nearly 27 times more bytes are necessary to transmit it over DoH (POST, GET) or DoH with JSON. The differences between individual DoH and DNS over JSON are not so significant, and each method exceeds the other method in one of the monitored statistics. Regarding payload size, the most efficient encrypting approach is DoH using the POST method. The GET method shows a 0.27% increase in payload size over the POST method, and DNS over JSON shows a 2.66% increase (based on median values). From the number of packets perspective, the most effective method is the DoH with the GET method. POST method utilizes 21.43% more packets, and DNS over JSON uses 14.29% more packets on average.

The DNS over JSON shows the shortest duration from the encrypted methods. HTTP method GET requires 0.86% more time, and method POST 9.38% more time than DNS over JSON. The time and complexity of creating queries can be the factors to consider for the applications; hence, the DoH using the GET and JSON methods would be the primary options.

Unsurprisingly, a single query response pair transferred using DoH requires more data and packets transmitted compared to DNS. DoH has to proceed with several phases of connection establishment. Compared to the traditional DNS, it can be seen that the DNS represents only a fraction of data transferred over the DoH channel. The overhead for a single query is significant and very costly but still can be used by some applications requiring limited domain name resolutions or actors hiding their activity. The initial overhead is amortized when more queries are transmitted over a securely established connection.

In addition, queries using HTTP/2 and HTTP/1.1 were measured to compare the behavior of single query DoH. The POST method was used to show the difference between the two versions. A slight difference can be observed mainly in the number of packets (see Figure 5.2e) as HTTP/2 frames and splits the message into separate packets, which also leads slightly longer duration of the HTTP/2 flows (see Figure 5.2f). However, both are almost the same regarding overall flow size (see Figure 5.2d).

5.2 DoH-Enabled Browser Analysis

Besides the implementations of the DoH protocol in operating systems, web browsers represent the most common applications currently implementing DoH. Applications no longer rely on the operating system domain resolution services but on their own. When enabled, a web browser has to discover an appropriate DoH server address.

The configuration varies depending on the browser. Browsers based on the Chromium code base have few predefined and hard-coded DoH resolvers, providing the possibility to set custom resolvers. Firefox provides default DoH options; the primary option is Cloudflare's resolver. Users also have the possibility to set up a custom resolver.

The browsers are discussed and analyzed since they represent the first and most stable implementation of the DoH protocol that is actively used. Compared to operating systems, the browsers can be much more easily controlled to generate DoH traffic while providing options for additional configuration. Therefore, those applications are used for data generation and testing.

5.2.1 Chrome

Chrome browser has implemented DoH resolution since version 78 (only experimental then). In the experimental version, the DoH was only activated when the system's DNS servers were set to one of the servers in a hard-coded table. Chrome released official support for DoH in version 83. Currently, the number of provided hard-coded servers [7] is increasing, and users can set their own server when needed. The setting of Chrome DoH is tightly related to the system settings. Chrome has supported DoH on Android OS since version 85.

DoH in Chrome works as follows: i) Chrome DoH client asks the system-configured DNS servers to resolve the following `dns.google` domain. ii) If a valid server address is retrieved, it establishes the HTTPS connection to the DoH server. iii) The DoH session is used for resolving all other domain names. The DNS message is Base64 encoded and sent as a query parameter in the HTTP/2 GET command. The answer is sent in a data stream of HTTP/2 in the DNS wire format.

Chrome contains a fallback mechanism if the DoH service replies error code. In this case, Chrome tries to resolve the domain using the system's DNS resolver. Chrome DoH also supports EDNS padding [90] which allows DoH clients and servers to increase the size of a DNS message, thwarting the size-based correlation of encrypted DNS messages.

5.2.2 Firefox

Firefox has provided a more tuneable environment for DoH since the beginning of the support of the DoH protocol. Users can set modes of DoH (force, try, off), HTTP methods (GET, POST) used for resolution, and a custom DoH server. The default provider is Cloudflare, but it is possible to specify any custom provider. When using the predefined provider, the browser attempts to resolve its domain name and selects one of the retrieved IP addresses to establish a DoH connection. In default settings, it queries `mozilla.cloudflare-dns.com` domain.

Upon successfully acquiring a DoH server address, the browser starts multiple connections with the server. One of the connections lasts for the entire lifetime of the web browser session. This connection is kept alive by using `PING` commands of HTTP/2 if necessary. The `POST` method sends DNS queries in the wire format. In the answer, the data stream also contains DNS wire format, which is indicated by `application/dns-message` content type. The requests coming from browser applications are uniform, either GET or POST. The change can be observed whether the server uses HTTP/2 or HTTP/1.1. Previously, the Firefox browser did not support EDNS padding as Chrome did. It was reported as an enhancement in Mozilla's bug report system [101], and later solved. However, the versions not supporting EDNS padding were used to generate all data in this work.

5.2.3 Other Browsers

Many browsers based on the Chromium code base such as Chrome, Edge, Opera and others contain the same DoH resolution implementation. The setting is similar to the one implemented in Chrome. Safari browser does not implement DoH.

5.3 Browser Page Load Impact of DoH POST and GET Methods

This section conducts an experiment that aims to measure the impact of different DoH methods on page loading. The two RFC 8484 defined methods are also compared to traditional DNS.

5.3.1 Methodology

Web browsers were used to generate DoH data. In experiments, the web pages were visited. The source of webpages was the Majestic Million dataset [87] maintaining the top million of visited websites on the internet. The Firefox browser was used in the experiments since it provides more configuration options (set HTTP POST or GET method for DoH resolution) than Chrome. The data were generated by browsers running in the Docker containers as depicted in Figure 5.3. The traffic is then captured on the Docker network interface.

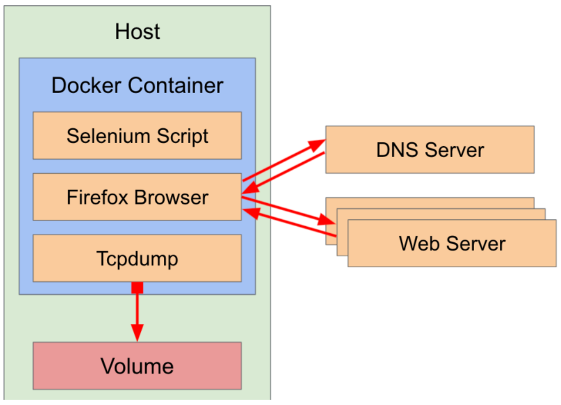


Figure 5.3: DoH generation browser.

The browser is prevented from running in headless mode to simulate the close-to-real browser behavior better. Instead, X virtual frame buffer is used. It implements the X11 display server protocol where all graphical operations are made in virtual memory without graphical output [34]. This mimics the browser's usual usage of the browser by a user. The Selenium script drove the browser operations.

The browser cache was disabled, and the DNS cache expiration was set to 0. This way, every domain has to be resolved and not acquired from the cache. The requests for the web pages are made separately. Before each request, the browser is opened, and after each request, the browser is closed. Using this operational pattern, the browser generates more DoH traffic. The generated data are part of the whole dataset [71]. Besides the captured traffic, HTTP Archive Veiwer (HAR) logs were recorded to obtain event logs from which the page load times could be retrieved.

The same infrastructure as in the previous experiment (see Section 5.1) was used. The measurements run simultaneously on separate machines. About $100,000$ pages were loaded by the browser. The same Cloudflare resolver for domain resolution was used for queries using DNS, DoH GET, and DoH POST methods. The measurement was repeated three times to provide more reliable results, reducing the influence of unexpected network conditions. The round trip times of the resolver for both DoH and DNS were similar. The round trip time was measured, and the average difference between DNS and DoH servers from our measurement point was $0.02ms$. Cloudflare's DoH resolver accepts HTTP/2 connections.

5.3.2 Results

The presented results focus on the page load times. The page load times are extracted from captured HAR log records. They represent a time period between *domainsLookupStart* and *domComplete* browser events. Figure 5.4 shows the results of the measurements towards the Cloudflare's resolvers presenting the distributions.

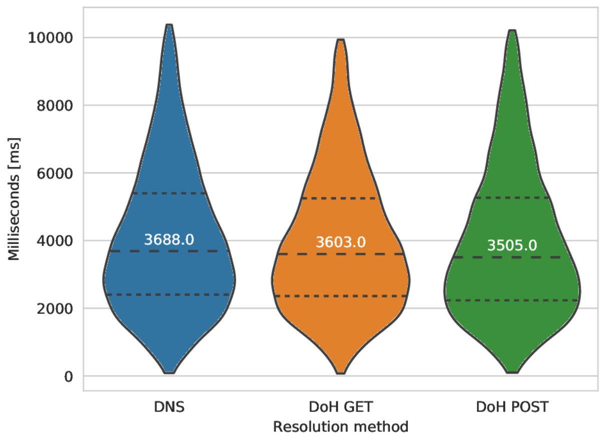


Figure 5.4: DoH page load time comparison of DNS, DoH using GET and POST towards Cloudflares server.

The distribution of the differences in all categories has a similar shape. The medians were chosen as the leading value in comparison due to the positively skewed distribution. Surprisingly, the DNS method appears to be the slowest, while DoH using the POST method is the fastest, with a difference of about $\sim 100ms$ between the DoH methods and $\sim 200ms$ between the slowest DNS and fastest DoH POST. According to the measurements, the difference between DNS and DoH POST reaches $\sim 5\%$. The difference is relatively low.

To confirm the results, DoH methods were measured again towards Google's DoH resolver. The methodology is still aligned with previous experiments. Figure 5.5 depicts the comparison, and a similar gap between medians of resolution methods as in the previous measurement is present. In both cases, the DoH POST method has slightly better performance in page load times.

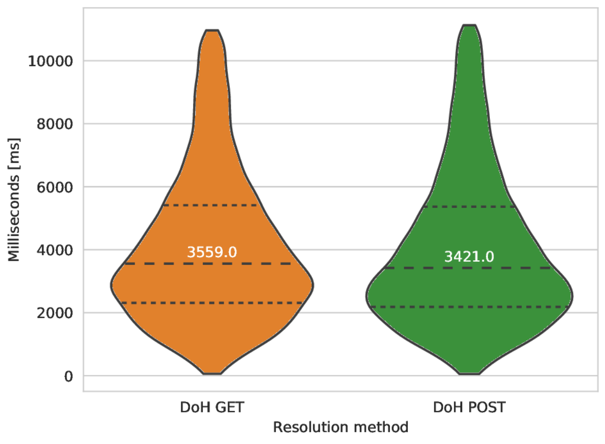


Figure 5.5: DoH page load time comparison DoH using GET and POST towards Google server.

The POST method is a winner in this experiment with respect to web page load latency. Also, POST requests are generally smaller than their GET equivalents. However, the results are given by the methodology where the DNS caching mechanism is disabled. In reality, domain name caching can reduce the resolution time, and the GET method may be more suitable since the responses to POST requests are generally not cacheable (unless a special response header sent which is not advised for DoH) [51]. Privacy Google's best practices for DoH [46] claim that using the POST method is more suitable for privacy-critical applications, where caching is undesirable.

5.4 Traffic Analysis of DoH Towards Multiple Resolvers

The browsers are applications implementing the client side of the DoH protocol. The client-side applications use minimal HTTP headers to produce DoH queries. If possible, they effectively utilize DoH over HTTP/2, capable of resolution also using HTTP/1.1. This section analyzes the characteristics of DoH communication, considering different DoH resolvers. Many providers currently deploy DoH servers in their installations using various DoH software, versions, configurations, operating systems, etc. The previous Chapter 4 covered the characteristics of various DoH resolvers. Moreover, the DoH resolvers are part of the HTTPS ecosystem, including load-balancers, reverse proxies, etc. Thus, to determine the real-world DoH traffic characteristics, all parts of the ecosystem should be considered.

The analysis aims to characterize DoH traffic by showing the similarities and differences of a DoH resolvers sample. Also, the difference between DoH and regular HTTPS communication is provided.

5.4.1 Methodology

The traffic used for the analysis is generated using the same methodology as in the previous experiment (see Section 5.3.1). A sample of servers covering different characteristics is taken from a well-known resolvers list. Moreover, the resolvers providing only HTTP/1.1 support are also chosen to show the differences between HTTP/1.1 and HTTP/2 DoH communication.

5.4.2 Results

The results are presented on the set of graphs showing the correlation between average payload size and packet number of flows. The correlation of those features was chosen for demonstrative purposes covering the differences in DoH communication.

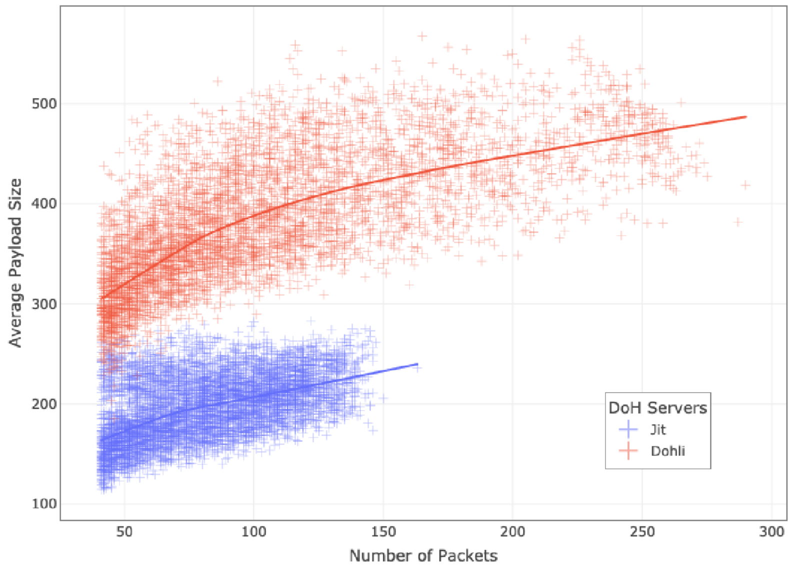


Figure 5.6: HTTP/1.1 DoH servers payload size to packets comparison.

Only a small portion of DoH resolvers exclusively operate on HTTP/1.1. Two such resolvers doh.li and jit were chosen to and the ratio of average TCP payload sizes to packet numbers is shown in Figure 5.6. Flow includes statistics from all packets within a flow; zero values, e.g., empty TCP acknowledgment packets, are included.

The observable difference is mainly caused by different HTTP headers sent in DoH responses generated by the resolvers. While the web browser (DoH client) always sends minimal necessary HTTP headers in requests. Each server sends responses containing various HTTP headers, including unnecessary ones, e.g., *X-Powered-By* as presented in previous Chapter 4. The HTTP/1.1 does not compress the headers sent in plain text, representing a significant overhead in each message. In most cases, the size of header lines exceeds the amount of DNS data. DNS resolution should be fast. Applications (primarily browsers) send multiple requests in a short period. The head-of-line (HOL) blocking should cause a considerable performance decrease in the case of HTTP/1.1. The browsers try to overcome the problem by opening multiple connections and sending requests in parallel connections. Unfortunately, the problem with HOL blocking is still present there.

The majority of the servers from the DoH list support HTTP/2. Figure 5.7 depicts the ratio of packet size to packet number in this case for HTTP/2 DoH servers. It can be observed that the average size of the packets is lower compared to the previous one utilizing HTTP/1.1, which is caused by the design of HTTP/2. The HTTP/2 performs HPACK header compression [110] mechanism in each request/response. Moreover, data and headers can be transferred in separated frames (also packets), which lowers the average payload size and increases the number of packets. One HTTP/2 flow can hold multiple streams, leading to an increased number of packets exchanged within a single flow. It can be seen that using this characterization, it is possible to distinguish different DoH resolvers.

Each connection always begins with a TLS handshake that has a similar characteristic for all communications and significantly contributes to the increased size/packet ratio, especially in short-flow characteristics. The TLS handshake is amortized near $1,000$ packets in a flow. Thus,

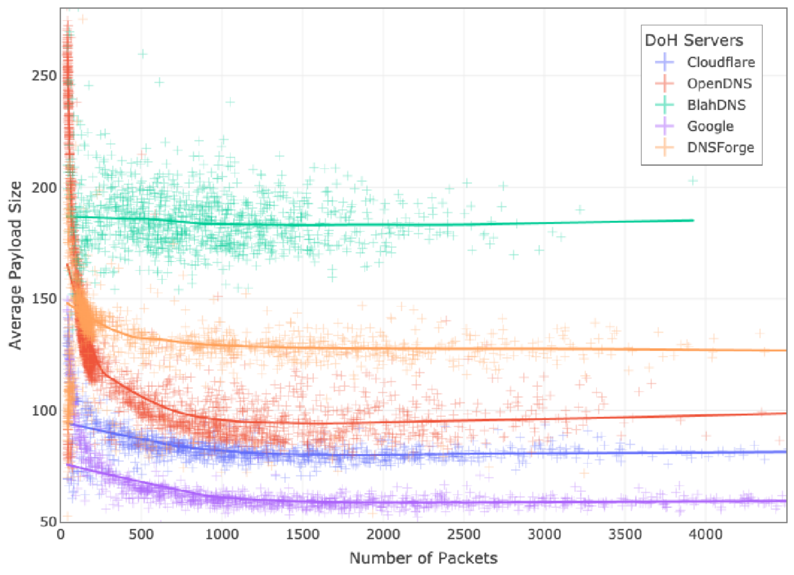


Figure 5.7: HTTP/2 DoH servers payload size to packets comparison.

for long-term DoH connections, corresponding curves converge to their specific constant average payload sizes.

Noticeable differences can be observed in the shape of curves for HTTP/1.1 and HTTP/2 cases. While DoH with HTTP/2 is decreasing average sizes the connections with HTTP/1.1 are increasing sizes up to some point. It can also be seen that the HTTP/1.1 connections are shorter in comparison to HTTP/2. Even the failed connections are included in the graphs which can cause the slump in short connections.

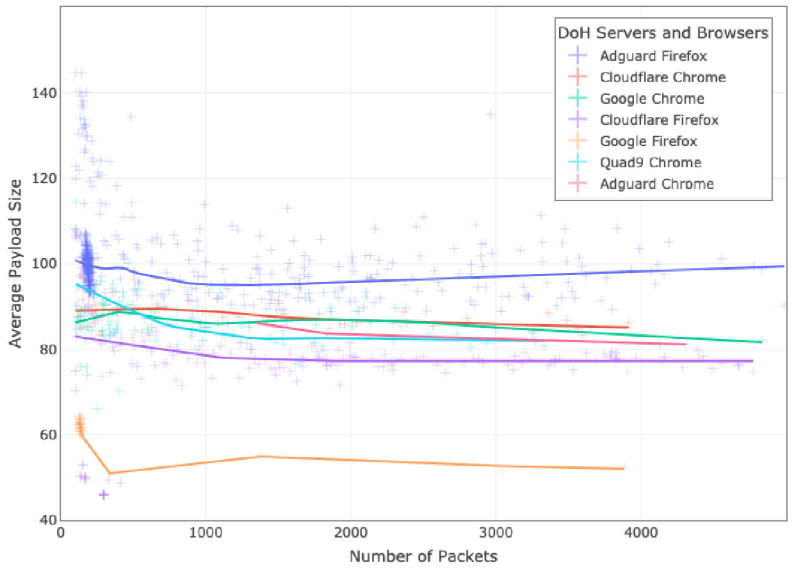


Figure 5.8: CIRA-CIC-DoHBrw-2020 dataset HTTP/2 DoH servers payload size to packets comparison.

To confirm the results data from another public dataset was examined. The most used *CIRA-CIC-DoHBrw-2020* dataset [96,97] was processed, and the data are shown in Figure 5.8. According to the dataset specification, data were generated from the Firefox and Chrome DoH communication with servers on HTTP/2. Only a benign part of the dataset was used. The dataset has fewer flows than the generated one but with a longer duration. Nevertheless, characteristics similar to the generated data can be observed. The difference between multiple resolvers using the same browser, as well as the difference between using Chrome and Firefox browsers towards the same resolvers, can be identified. Similar characteristics can be observed. Moreover, it is shown that the Firefox browser has a lower ratio than Chrome when using the same resolvers. It can be attributed to Chrome's implementation of EDNS padding.

DoH network traffic of different web browsers and DoH servers can be distinguished by using a simple payload size to packet number ratio. The protocol analysis shows that EDNS extension and HTTP header size are the most significant contributors to the characteristics.

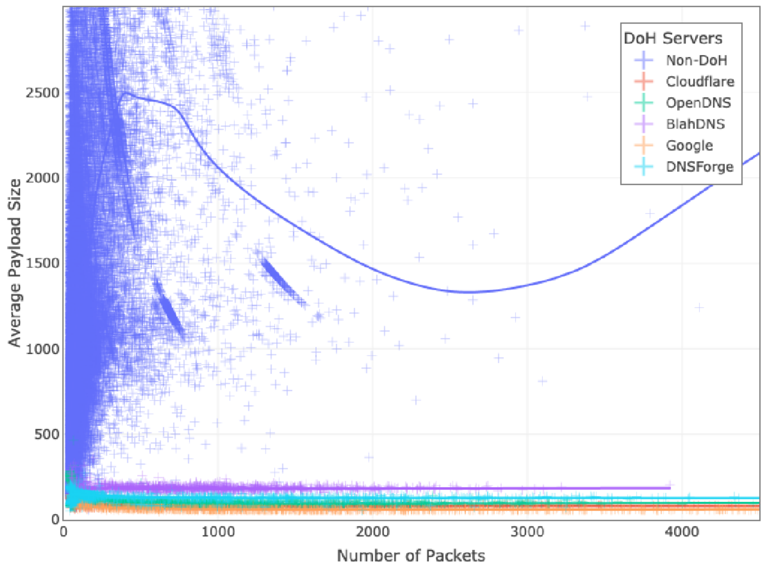


Figure 5.9: HTTP/2 DoH servers payload size to packets comparison together with the rest non-DoH HTTPS traffic. The graphs include trend lines that support the trends useful for showing the trends of DoH resolvers.

Filtered DoH traffic can be further put into contrast with the rest of the HTTP traffic provided in datasets. Figure 5.9 depicts both DoH traffic generated towards multiple DoH resolvers and the regular HTTPS traffic, all in blue color. The traffic is generated as described in the methodology; single web pages were visited, and connections contain various transferred assets. Mostly flows carrying images, short presentation videos, CSS, JavaScript files, data, and other standard documents. Hence, the regular HTTPS traffic mostly lacks longer streams and interactions, as can be observed. The depicted HTTPS traffic contains larger packets (reaching up to 64 KB), which is caused by enabled TCP offloading in virtualized infrastructure. This fact does not change the validity of DoH characteristics but provides a view into regular HTTPS traffic in one of the possible environments. The graph is zoomed in for better coverage of the difference between DoH and the rest of HTTPS rather than the full picture of HTTPS traffic.

Figure 5.10 provides similar observation, except that it comes from the *CIRA-CIC-DoHBrw-2020* dataset data. The dataset is generated differently. Hence, the characteristics of the non-DoH traffic are different. During the generation, longer interactions were made, and the non-DoH traffic got higher diversity and behaved more realistically. However, it can be seen that the DoH can still be distinguished considering the feature of payload size to packet number ratio.

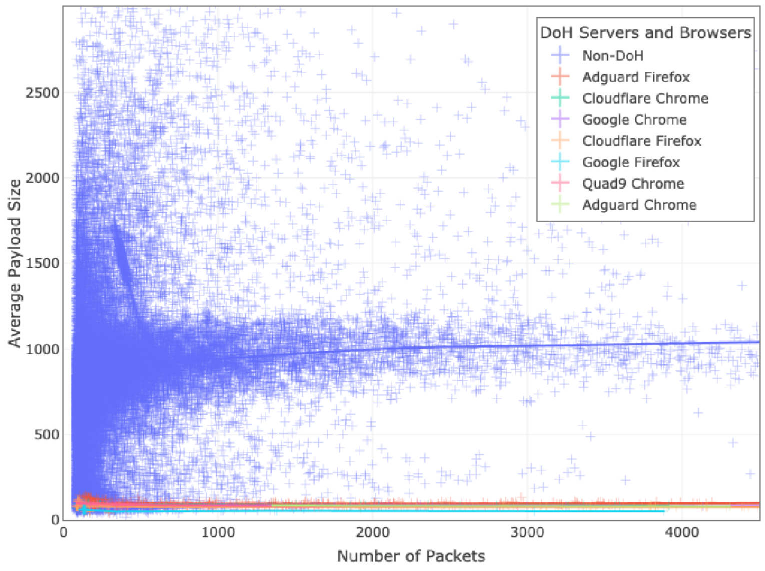


Figure 5.10: CIRA-CIC-DoHBrw-2020 dataset HTTP/2 DoH servers payload size to packets comparison together with the rest of non-DoH HTTPS traffic.

Additionally, compared to regular HTTPS, the DoH creates more equal connections regarding packets sent and received. Small DoH requests have their small counterpart responses. Despite the stream multiplexing (supported in HTTP/2), each DoH request/response is mapped into a single network packet [61], which defines the traffic shape of DoH connections as demonstrated. Apart from initial handshakes, both TLS and HTTP/2, the DoH connection is usually long-lasting and contains smaller packets that are sparsely distributed in the connections. Nevertheless, such traffic shape is also common for other HTTP-based APIs, and those are particularly challenging to distinguish from DoH. Browsers create a DoH connection at the startup and then use it till it shuts down. This behavior resembles other applications, such as operating systems and proxies implementing DoH protocol.

5.5 Discussion

Results provided in this section cover three related DoH areas. The first, single query DoH producing short DoH connections shows the increased overhead of the protocol to achieve the same results over an encrypted channel. The traditional DNS has a lower response time caused by the overhead of DoH, which is aligned with the results of previous works [54, 55]. Single DoH query in connection takes up to 14 to 17 packets. The HTTP2 caused an increase of an additional ~ 11 packets, reaching 28 packets. It is expected that such flows can be produced by the applications requiring only single resolution, scripts, or malware applications trying to hide their presence in the network.

The second area brought the result of the differences between DoH HTTP POST and GET methods. The resolution using the DoH POST method produced the lowest page load times. Moreover, the DNS was outperformed by the DoH in page load times. Similar results were presented by Hounsel et al. [56] where the DoT resolution achieved better results than traditional DNS.

The results presented in this chapter that are the most beneficial for designing DoH detection methods covering the behavior of DoH traffic comes from the browsers representing one of DoH utilizing applications and their traffic analysis. The browsers create long-lasting connections and use them for DoH resolution. The browsers choose HTTP/2 if provided but can operate using HTTP/1.1.

Only a few resolvers provide HTTP/1.1 only, as presented in previous Chapter 4. When HTTP/1.1 is used, the browser utilizes multiple connections for resolution. Otherwise, a single long-lasting HTTP/2 connection is used, and other started connections are terminated. The HTTP/2 connections separate requests and responses into single network packets [61]. Moreover, those are separated from transmitted headers.

Additionally, the traffic analysis showed that the DoH flows towards different resolvers can be distinguished. The amount of HTTP headers and EDNS padding are the factors that influence the packet sizes the most, which confirms the observations from the previous chapter. The DoH has a different traffic shape than most of the HTTPS traffic. It creates long-lasting connections, transferring small packets with an almost equal amount of packets sent and received due to small separated packets carrying request/response communication. The discovered characteristics of the browsers and previously discussed resolvers give a lead for creating more comprehensive datasets for the detection task than the most popular *CIRA-CIC-DoHBrw-2020* dataset. According to the analysis, better quality datasets should include more variety of traffic generated by multiple applications such as browsers and various DoH resolvers, both using different amounts of HTTP headers as well as implementing various EDNS padding strategies.

The unveiled characteristics show potential discriminators that can be used for designing robust feature vector used as an input into the machine learning method, creating a core of the DoH detection. Moreover, as the analysis showed, different resolvers can be distinguished from the traffic; the design of the features should focus on covering DoH traffic generally rather than crafting specialized features, forcing the method to learn specific DoH resolvers characteristics and limiting the generalization. Finally, the observed characteristics show potential to be covered by information provided in broadly used Netflow telemetry.

Chapter 6

Datasets

Despite the immense amount of published works focusing on DoH and DoH tunnel detection in recent years, the amount and quality of available datasets are very limited. Some works created and used their own datasets but did not published them, and some published them, including *CIRA-CIC-DoHBrw-2020* created by Montazerishatoori et al. [97] and dataset created by Vekshin et al. [128]. Those published were reused by other researchers in their works. Most of the works use the *CIRA-CIC-DoHBrw-2020* dataset, which was also studied in the previous chapter. The *CIRA-CIC-DoHBrw-2020* dataset contains a limited amount of DoH traces towards only four resolvers creating similar characteristics compared to the traces, that can be easily generated as seen in the same section. The other dataset created by Vekshin et al. [128] contains only two resolvers. The limited available datasets show the need for the creation of more comprehensive datasets covering a wide variety of DoH traffic characteristics.

Collected data prepared in the form of a described dataset are an indispensable part of any data mining and machine learning task. This chapter takes the already covered knowledge presented in previous chapters and uses it to create and publish more comprehensive datasets than the currently available ones. The datasets presented in this work were created and designed to propose a DoH detector and measure its performance.

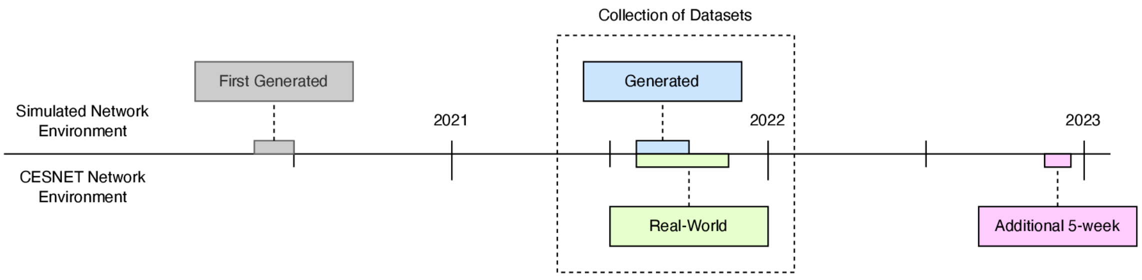


Figure 6.1: Timeline of datasets creation.

As the first attempt, clearly artificially generated dataset was created [71] to be used in the process of the DoH detector proposal presented in this work. The dataset is described in Section 6.1. Later, the methodology of artificial creation was improved, and a new, more comprehensive dataset containing data from the CESNET2 ISP network was published [65]. This dataset is covered in Section 6.2. This dataset was later extended [70] by an additional 5-week capture in the same real CESNET2 ISP network to measure the time data drift of DoH detection models. This additional dataset is presented in Section 6.3. Figure 6.1 depicts the timeline of datasets created and used in this work together with their summary presented in Table 6.1.

Table 6.1: Summary of datasets created and used in the thesis.

Dataset	**Published**	**Type**	**Resolvers**	**Clients**	**Size**
First Generated	[71]	Generated	11	1	55.5 GB
Collection of Datasets	[65]	Generated	16	2	250.5 GB
		Real-World Captured	Unknown	Unknown	179 GB
Additional 5-week	[70]	Real-World Captured	Unknown	Unknown	52 GB

6.1 First Generated Dataset

Other published datasets are limited in the diversity of traffic and in the amount of DoH traces. Moreover, the datasets were published later in 2020, along with the work that used them and was not present previously. Hence, new datasets needed to be created to study the DoH. The first generated dataset designated for DoH detection was created to cover the variability of possible traffic differences mainly caused by DoH resolvers. The dataset contains traffic generated towards a randomly chosen sample of 11 DoH resolvers from the well-known resolvers list. This dataset was published later [71] together with other generated traffic covering previous experiments presented in Section 5. Similarly, two other published datasets were produced and published at the time of this dataset generation, one by Vekshin et al. [128] together with their detection method [129] and the second mostly used in other works by MontazeriShatoori et al. [96] together with their detection method [97]. However, compared to the dataset presented in this section, their datasets were generated only towards a small number of resolvers mostly covering big effective resolvers; the first contains traffic towards two and the second towards four resolvers.

6.1.1 Methodology

The Firefox browser had one of the first implementations of the DoH protocol, and it provided the most configuration options, such as whether the POST or GET method should be used for resolution, modes of DoH enforcements, and whether to use additional headers. Firefox also supported DoH on Linux, which simplified the option for automation and deployment of the generation process; Chrome did not support DoH on Linux, only on the Windows operating system.

To generate the DoH traffic, the Firefox browser was used in the Docker container. Using the Docker container does not influence the traffic with specific properties; however, it simplified the generation process automation and provided an isolated application environment. The platform utilized in the data generation process using Firefox browser was Supermicro SuperTwin2 6026TT-TF server equipped with eight Intel (R) Xeon E5520 @ 2.26 GHz. The cluster consists of 4 nodes. The nodes were equipped with 48 GB RAM and 16 CPU cores. The platform was directly connected to the Brno University of Technology network. Network traffic was captured directly in the container using the *tcpdump* command-line tool.

Python and Selenium libraries were used to automate the generation process. To be able to provide more realistic behavior of the browser, we used an X-virtual frame buffer, which implements the X11 display server protocol where all graphical operations are made in virtual memory without graphical output [34]. The browser then works with a standard graphical output even within the container.

A sample of 11 chosen resolvers was taken from the curl community list of known DoH resolvers [126]. Firefox had disabled the local web cache during web page access to get more network traffic samples. The process consists of opening the browser, fetching the website, timeout, and waiting for the loading of the website, followed by browser closing. The schema of this generation is depicted in Figure 6.2. This process was repeated 200 times for DoH POST and GET

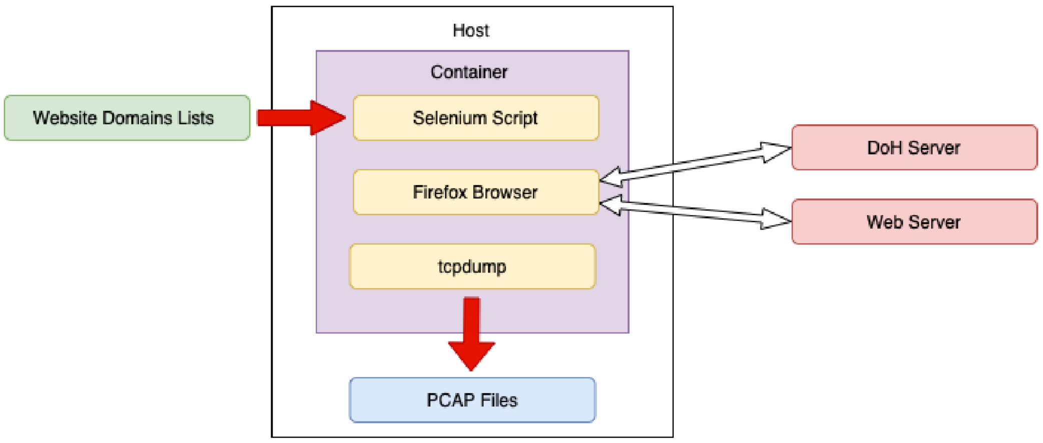


Figure 6.2: Firefox browser traffic generation schema.

methods and for each of the 11 DoH resolvers. Together, 4400 different websites were visited taken from the Majestic Million dataset [87].

6.1.2 Data Description

The dataset consists of network traffic generated by the Firefox browser. In total, 4400 web pages were visited, and services of 11 resolvers were employed. The Firefox browser was enforced to make 200 fetches by each of the supported DoH HTTP methods.

The dataset consists of full-packet captures of DoH and regular HTTPS traffic generated solely by the Firefox browser application. Full-packet captures are in standardized PCAP format [48], which is a default for *libpcap* library and broadly supported by network analysis software, e.g., Wireshark[1], *tcpdump*[2] or IDS Suricata[3]. The PCAP files do not provide a separate DoH and regular HTTPS traffic. The relevant traffic belonging to each category can be distinguished by the IP addresses of DoH resolvers shipped together with a dataset. Since specific DoH resolvers were used, the IP labeling provides an accurate way to separate the DoH data. Table 6.2 summarizes the statistics of the dataset.

Table 6.2: Total stats of captured data.

Name	**Value**
DoH resolvers	11
DoH clients	1
Total web pages visited	4400
Total data size	55.5 GB
Connections	761 K

[1]https://www.wireshark.com
[2]https://www.tcpdump.org
[3]https://suricata.io

6.2 Collection of DoH Datasets

The new, more comprehensive dataset that overcome the limitations of previously published datasets was created and published [65]. The dataset is denoted as a collection since it consists of multiple DoH traffic data created by multiple ways and is presented in this section[4].

The collection of datasets consists of captures of DoH traffic and regular HTTPS traffic that comes both from controlled and real-world environments. The real-world dataset was collected in an ISP backbone network servicing half a million users. The other generated dataset with DoH captures consists of traffic generated by several techniques and two browsers towards 16 selected DoH servers of various implementations and with different configurations. The generated part contains 64 000 web page accesses and related DoH communication. The datasets aim to provide a more comprehensive sample of DoH traffic as observed in real networks and various available implementations and configurations.

6.2.1 Methodology

The datasets contain all DoH, and regular HTTPS captured flows. The common attribute of the whole dataset collection is that the data are captured in a pure form, where the process and all necessary methodologies are covered in the following sections. The dataset consists of the generated network traffic denoted as *Generated Data* covered by Section 6.2.2, and traffic captured from a real network referred to as *Real-World Data* presented in Section 6.2.3.

Since the data are filtered only for TCP port 443 covering DoH and the rest of HTTPS traffic, the dataset may contain all sorts of DoH and non-DoH HTTPS flows, e.g., long, short, successful, failed, and even containing only one packet in the way they appeared during the capturing process. No additional filtering that would lead to information loss except those mentioned in the following two sections was performed.

6.2.2 Generated Data Methodology

As opposed to the first generated dataset covered in the section above, this dataset uses two DoH-enabled web browsers for making web requests. In addition to Firefox, Chrome browser was also used. The Chrome browser did not provide the same configuration options as Firefox, but it enforced EDNS padding, which Firefox did not implement at that time. Together, they have the most advanced DoH implementations covering client-side variability. Since the primary target is DoH and Non-DoH traffic generation, all possible settings that can affect the DoH traffic characteristics (such as packet sizes) generated by those applications should be covered.

The *generated data* samples were created by visiting a collection of websites that are part of the Majestic Million list [87]. The URLs were taken from the beginning of the list, and each website was visited by one of the browsers. None of the URLs is used more than once.

DoH traffic characteristics are mostly influenced by client-server interaction. As mentioned earlier, the resolvers vary in their implementations (different software and versions) and different configurations. Moreover, the HTTPS ecosystem on the server side can consist of proxies, firewalls, load balancers, and other systems that impact network communication. To cover various DoH traffic patterns, a different sample of 16 DoH servers was chosen to generate datasets. The 16 DoH servers were chosen to cover various DoH traffic patterns as described in previous chapters (see Chapter 4 and Chapter 5).

[4]This section is based on the published article [65] presenting the published dataset by the author.

The whole traffic generation process took place at the Brno University of Technology, Faculty of Information Technology. The machines were directly connected to the university network.

Firefox Browser Traffic Generation

The Firefox browser in version 76.0.1 was used in the Docker container. The methodology is similar to the case of the previously generated dataset presented in Section 6.1.1. The process, configuration, and hosted hardware are the same. In addition to the previous configuration, the container was restricted to use 4 CPUs and 4 GB of RAM at maximum.

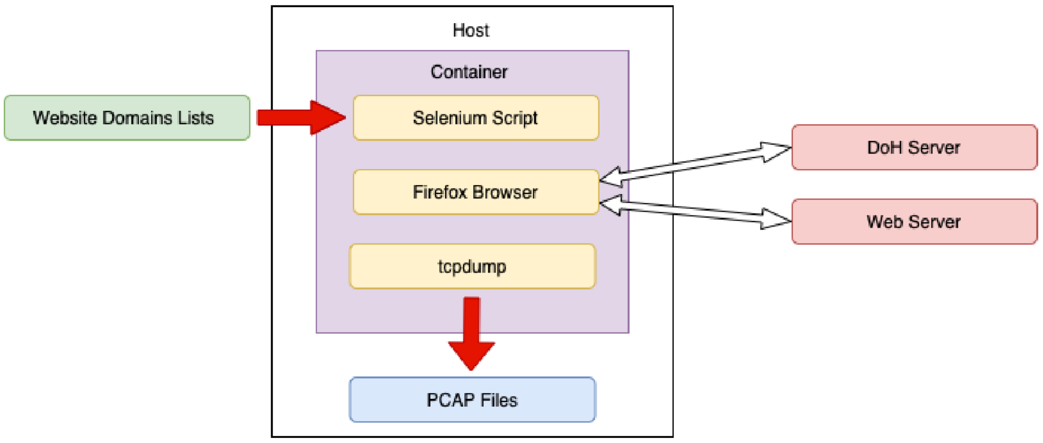


Figure 6.3: Firefox browser traffic generation schema.

Firefox had disabled the local web cache during web page access as it proved to generate more network traffic samples. The browser was opened and closed between each page visit. The schema of the generation process is also the same as depicted in Figure 6.3. However, the number of fetched websites is much higher. The process was repeated 1000 times for DoH POST and GET methods for each of the 16 DoH servers. Together, 32 000 different websites were fetched.

Chrome Browser Traffic Generation

Since the used version of Chrome browser (94.0.4606.81) did not support DoH on Linux OS at the time of data generation, the same generation environment as in the case of Firefox could not be reused. The Chrome browser generation took place on a separate machine running Windows OS without any virtual environment. The machine was equipped with 3rd generation Intel Core i5, 8 GB RAM, also connected directly to the same university network as in the case of Firefox.

The traffic generation process was similar to Firefox. It consisted of opening the browser with the given website, timeout for website loading, and followed by browser closing. This generation is shown in the supportive schema in Figure 6.4. A Python script and Selenium handled the automation as in the previous case, except that the resolver was set manually for each resolver. The same 16 DoH servers as in the case of Firefox were used. Chrome does not allow enforcement of HTTP GET or POST querying methods for DoH. Hence, the amount of fetched websites is similar to the Firefox case: 2×1000 only with the default Chrome method used. In total, 32 000 different websites were fetched.

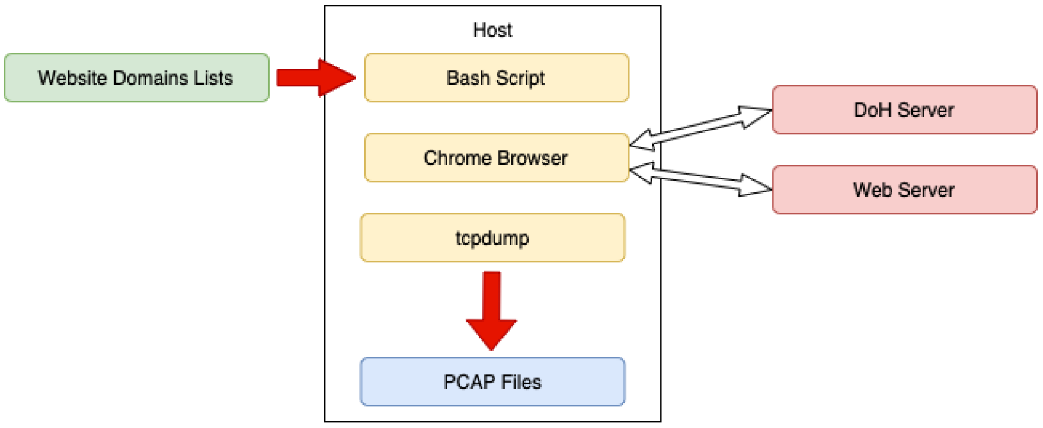


Figure 6.4: Chrome browser traffic generation schema.

6.2.3 Real-World Data Methodology

The *Real-World data* of the dataset were captured on the monitoring points of the CESNET organization, which is the Czech National Research and Education Network operator. CESNET operates a backbone network infrastructure called CESNET2, which half-million users use. CESNET2 provides internet connectivity to universities, campuses, research centers, schools, hospitals, and some government offices. The topology of the CESNET2 backbone network is depicted in Figure 6.5.

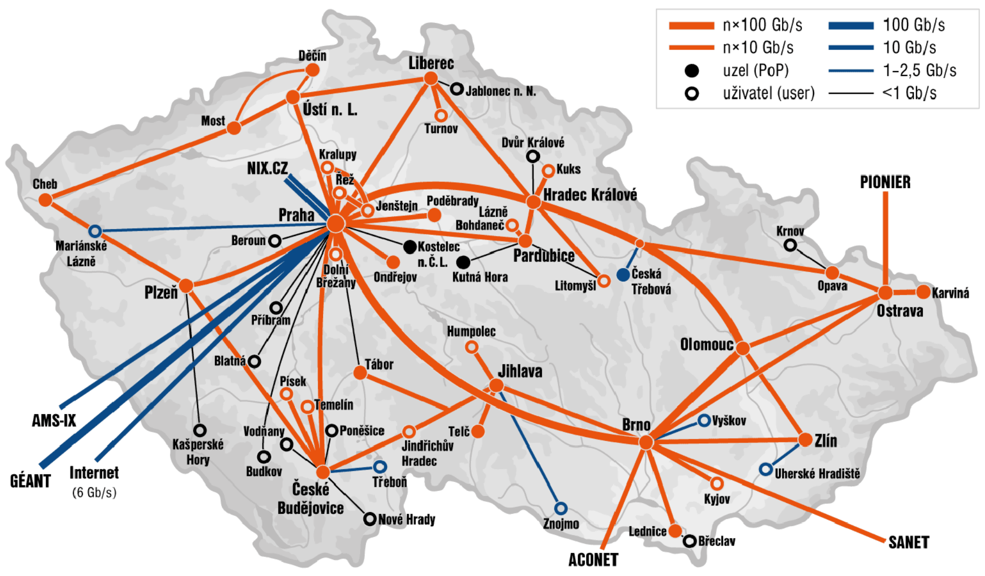


Figure 6.5: The topology of CESNET2 network [65].

The six monitoring points are located at the perimeter of the CESNET2 network in three locations in the Czech Republic — Prague, Brno, and Ostrava. Each of them is connected to one or multiple 100 Gbps lines via passive optical TAP. Since the packet capture is distributed across multiple points and locations, there could be small time deviations even if all the points are synchronized via Network Time Protocol (NTP). The time shift between monitoring probes does not affect connections that are routed symmetrically, meaning that all packets belonging to the one TCP connection pass through the same monitoring point.

The Real-World Data may contain small shifts in interarrival time values between packets in the opposite directions of some flows that are routed asymmetrically on the CESNET2 network — packets in the same direction are always routed via the same monitoring points, so no shift is present there. However, the timestamp inaccuracies are very small, and they do not cause any significant disturbances in packet order within the connection (HTTPS requests are always before HTTPS responses) and have a very low impact on the timing. Hence, this does not lower the data quality by any means. Moreover, since the distributed monitoring infrastructure is common in large-scale network monitoring [53], these data imperfections are viable for the dataset, creating the detection algorithms prepared for such deployment.

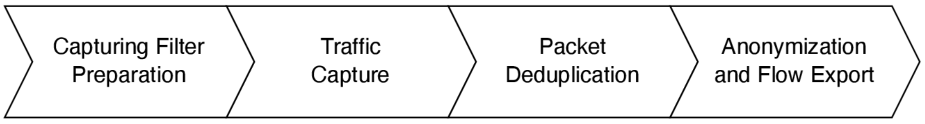


Figure 6.6: Real-World Data creation process.

As it can be seen in Figure 6.6, the creation of Real-World data can be divided into four steps: *(i)* Capturing filter preparation, *(ii)* Traffic Capture, *(iii)* Deduplication, *(iv)* Anonymization. Each step is described in the following sections.

Capturing Filter Preparation

The monitoring points can process 100 Gbps traffic; however, it is not possible to capture all ongoing traffic due to bandwidth limitation and the writing speeds of the storage, which would result in missing packets in the dataset. Therefore, it was necessary to create a capturing filter with this limitation in mind and reduce the amount of captured data to certain IP subsets so that all packets can be captured. Since the use of DoH is still small, all DoH traffic could be captured by filtering based on the DoH resolver's list taken from the monitoring of well-known DoH resolvers (see Chapter 4).

In the case of HTTPS traffic, the capturing was limited to the selected /24, and /22 IPv4 address ranges assigned to the university campus. Additionally, the whole capturing process was monitored for packet drops to ensure that all transmitted packets in filtered connections were saved on the disks and the points were not overloaded; thus, no artificial packet drops were added, and all the original data were captured, and hence the quality was guaranteed.

The capturing filters remained unchanged during the whole process of capturing. No changes were observed in the monitoring of resolvers during that period. All the data come from the same subset of IPs monitored for the whole period of data capture, ensuring the consistency of the data.

Traffic Capture

The traffic was captured on the previously mentioned distributed metering points in parallel for a certain period and then collected and merged into a single PCAP file using *mergecap* utility[5].

Packet Deduplication

Since a backbone ISP network can also carry transit traffic, there is a chance of capturing duplicate packets on distributed monitoring infrastructure. The deduplication was performed with editcap[6]

[5] https://www.wireshark.org/docs/man-pages/mergecap.html
[6] https://www.wireshark.org/docs/man-pages/editcap.html

utility with a time window of 10 ms. Thus, if there are two or more duplicate packets within 10 ms time window, only the first packet will be preserved.

Anonymization and Flow Export

Since data from real ISP backbone lines are used, sensitive information has to be removed from the final data. The anonymization of captured traffic was performed automatically; thus, nobody could view or analyze raw source data.

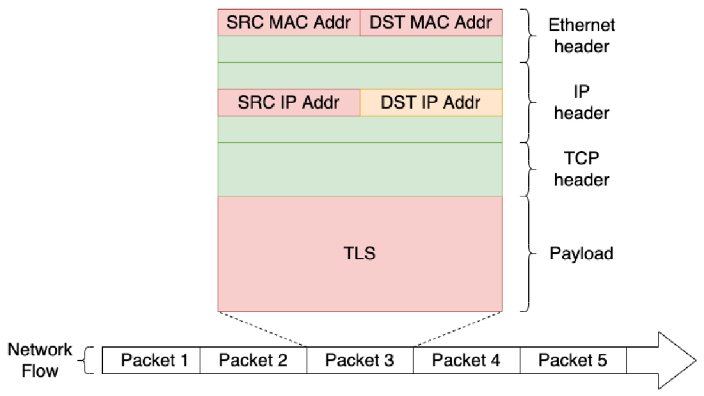


Figure 6.7: Structure of the packet. The **red fields** are always anonymized, **orange field** is anonymized for other IP addresses than the known DoH servers, and **green fields** are always left intact.

At first, all addresses (IPv4, IPv6, and MAC) were anonymized, except the IP addresses of DoH resolvers, which remained unmodified. The anonymization process replaced each address by a part of its SHA256 hash (the first N bytes, where N is the length of the corresponding address). The hashing algorithm ensures that a particular address is always mapped onto the same anonymized value.

After the flow export, the payload from all packets was also anonymized by substituting every byte with the letter 'X'. Removing packets' payload does not damage packet metadata; however, it ensures the privacy of the real users.

6.2.4 Data Description

The datasets consist of labeled HTTPS traffic. The traffic is either the DoH or regular HTTPS communication. The primary focus of this dataset collection is to provide variable and representative DoH traffic obtained from communication with many available DoH services. The collection of datasets contains DoH and HTTPS packets in the total amount of 430 GB of raw binary data. Full-packet captures are in standardized PCAP format and can be processed by regular network traffic processing tools and libraries.

Generated Data

The datasets with generated traffic were created in a controlled environment to provide samples of DoH communication for different existing implementations in operating systems and web browsers. The traffic originates from DoH-capable browsers Firefox and Chrome. Browsers access a large collection of web pages, and both DoH and HTTPS communication are captured.

A sample of 16 different DoH servers was chosen to include a diversity of traffic characteristics related to domain name resolution. For each DoH server, traffic was generated by fetching 2000 websites on each web browser. In total, 64 000 website requests were generated, and corresponding HTTPS communication was captured. Traffic samples generated by the Firefox browser contain packets up to 64 KB because the browser was executed in a virtual environment with TCP offloading enabled [131]. It provides more variability in the data, influencing only connections transmitting bigger data, not influencing DoH connections. Hence, it creates a complement to variability together with traffic generated by Chrome with TCP offloading disabled.

Table 6.3: Total statistics of Firefox and Chrome generated data.

(a) Total statistics of Firefox generated data.

Name	**Value**
Total data size	131 GB
DoH resolvers	16
Connections	~1.25 M

(b) Total statistics of Chrome generated data.

Name	**Value**
Total data size	119.5 GB
DoH resolvers	16
Connections	~723 K

Table 6.3a shows summary metrics of all raw generated traffic data by Firefox across all generated datasets in the collection. Table 6.3b depicts summary information about all traffic data that comes Chrome browser across all generated datasets in the collection. The difference in the number of connections can be caused by the generating process discussed later and browser-specific implementation.

The Chrome browser implements EDNS padding, slightly impacting the DoH flow characteristics when compared to the Firefox DoH flow, which yields an increased diversity of DoH traffic (POST and GET methods).

Real-World Data

The *real-world* data contains DoH and web-based HTTPS communication transmitted over standard HTTPS port 443. The real-world traffic captures were anonymized to protect the privacy of real users (packets have no payload, clients' IP and MAC addresses were hashed). However, anonymization does not pose a major limitation for data usability since it maintains the original timestamps of the packets and other important network traffic characteristics. The DoH captures contain only DNS over HTTPS communication, obtained by filtering connections using the IP addresses of known DoH resolvers.

Most DoH connections involve Google DoH resolvers, and the five most popular resolvers represent more than 93% of all DoH traffic, as depicted in Figure 6.8.

Table 6.4: Total stats of captured data.

Name	**Value**
Total data size	179 GB
Total time	~10 Days
Connections	~5.4 M
Number of unique Client IP addresses	116 263
Number of unique Server IP addresses	9343
Number of unique DoH Resolver's IP addresses	142

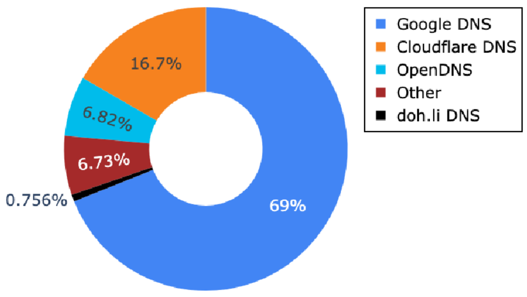


Figure 6.8: DoH resolvers' share.

The importance of the *real-world* dataset is that it contains the HTTPS traffic of thousands of genuine users. The Table 6.4 shows the overall statistics of the captured dataset. It should be noticed that there are 116 263 unique client addresses in the network.

6.3 Additional 5-week Real-World Dataset

Additional capture was made to provide data mainly for the demonstration of detectors' data drift. To provide insight into long-term and short-term usability and degradation of proposed DoH detectors in a real network environment. The data were captured in the same environment with the same setting as in the case of the previous capture on the CESNET ISP network.

6.3.1 Methodology

The methodology follows the same capturing, processing, and anonymization procedures as described in Section 6.2.3. However, the times of capture are different. The capture was automatically performed every Monday evening for five weeks between 28 November and 26 December 2022. The procedure was stopped during Christmas since the traffic was captured from the university campus, and the activity on the network went down during that time.

6.3.2 Data Description

The dataset consists of 5 weeks of separated labeled DoH and regular HTTPS traffic. The dataset is again in the form of full-packet capture in standardized PCAP format [48] that any of the standard processing tools can process. Table 6.5 summarizes the dataset statistics.

Table 6.5: Total stats of captured data.

Name	Value
Total data size	52 GB
Total time	5 Weeks
Connections	~503 K

Chapter 7

Proposed DoH Detection Approach

This chapter covers one of the main goals of this thesis and presents the proposal of the DoH detector that is motivated by mass DoH employment. The detection method proposal was published in article [66] by the author, and this chapter is based on the results presented in the published paper. DoH is widely misused due to its stealthiness in the network. Compared to other encrypted DNS approaches, DoH is designed to blend into regular HTTPS traffic, making its reliable detection challenging. As discussed in Chapter 4, DoH cannot be reliably detected by lists of IP addresses and domain names due to multiple small and private DoH resolvers and changes in their deployment on the Internet. Therefore, researchers already use the statistical properties of connections in combination with Machine Learning (ML) to recognize DoH and achieve an accuracy of more than 99%. The high accuracy of these detectors comes with the requirement of additional information about the connections, such as individual packet lengths [129] or median and mode of packet sizes [10, 12, 21, 94, 97, 138]. Extraction of these traffic features is usually not supported by standard network monitoring tools that can be deployed on high-speed networks, which significantly limits the possibility of their deployment.

Additionally, survey [62] also identified the difficulty of detecting single query DoH, which needs to be addressed. Since DoH creates a DNS Application Programming Interface (API) over HTTPS, short connections have very similar characteristics to other HTTPS API requests. Moreover, other no-payload parameters, such as HTTP headers or the size of the HTTP/2 preface, influence the connection shape much more significantly than the DoH payload itself, as shown in the previous chapters, making the classifier fit specific server settings. Nevertheless, previous studies did not focus on short DoH connection [62] problems, even though they significantly influence the accuracy of DoH detection in the real world.

Given that none of the existing proposals provide a satisfactory solution for reliable DoH detection in the real environment. Particularly, they use hard-to-compute features and ML-based classifiers with low but still unacceptable false positive rates. Low false positive rate is important property of network detectors that make large number of inferences. Even 0.001 false positive rate is not enough when deployed on a network with throughput of 1000 flows per second, since it produces one alarm every second overwhelming the security personnel with a high amount of false alarms [5]. The proposed DoH detector can work with the standard flow data source, making it deployable to almost any flow monitoring infrastructure (from local area networks to high-speed backbone lines) while still maintaining a high accuracy of over 99.9%. The standard flow statistics can be obtained from all NetFlow capable appliances, such as switches[1], firewalls[2], or specialized flow monitoring probes such as Flowmon[3]. To overcome the information-limited standard flow telemetry data source, a combination of heterogeneous classifiers is used—IP-based, machine learning, and active probing detectors. The industry-proven IP filtering is based on the dynamic list of identified DoH servers, automatically adjusted when the new DoH server is identified in the monitored communication. The ML-based DoH classifier is used to identify candidate DoH servers by performing an online analysis of flow data, which are then confirmed by the active probing detector, resulting in almost zero false positives. Moreover, the proposed multi-classifier approach can detect even short DoH connections, which are neglected by existing works.

7.1 Used Datasets

High-quality datasets are crucial for designing a reliable and accurate network classifier. The de-facto standard dataset created for DoH detection is called *CIRA-CIC-DoHBrw-2020* [96], and it is used by the majority of related works. The *CIRA-CIC-DoHBrw-2020* contains DoH and regular HTTPS traffic in the form of extended flow records and PCAP files. The traffic, either DoH or regular HTTPS, was generated by Firefox and Chrome browsers, querying only four DoH resolvers at varying network speeds.

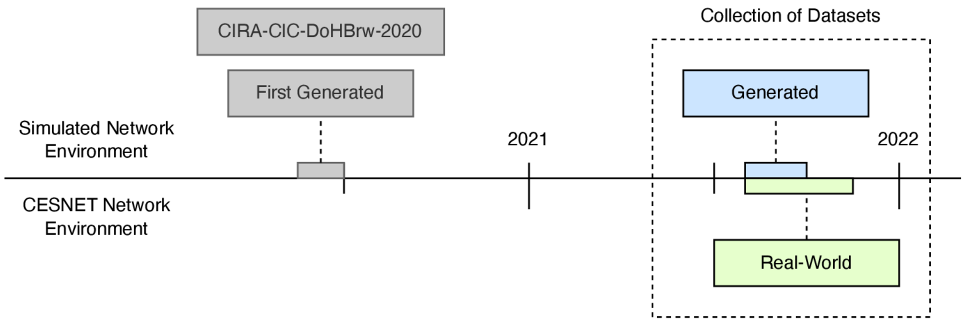


Figure 7.1: Used datasets timeline.

Nevertheless, the behavior of DoH resolvers and browsers significantly differ as stated in previous Chapter 5; thus, the four resolvers in *CIRA-CIC-DoHBrw-2020* cover just a fraction of DoH traffic shapes that can be observed on the internet. To overcome the limited number of DoH resolvers in the *CIRA-CIC-DoHBrw-2020* dataset, the first generated dataset described in Section 6.1 is also used. The DoH queries are generated against 11 different DoH servers, making the dataset more comprehensive.

Packet captures (PCAP files) from both *CIRA-CIC-DoHBrw-2020* and the first generated datasets was merged into one *Design dataset*, which was used for detector design. A similar approach of merging two datasets to provide a more comprehensive one was also used in other published works [72, 93]. The packet captures were then processed by the NetExP tool, aggregating the packets into standard bidirectional flow records.

[1]https://www.cisco.com/c/dam/en/us/td/docs/security/stealthwatch/netflow/Cisco_NetFlow_Configuration.pdf
[2]https://www.cisco.com/c/en/us/td/docs/security/asa/special/netflow/asa_netflow.html#bidirectionalflows
[3]https://www.flowmon.com/en/products/appliances/probe

Table 7.1: Overall information about used datasets containing the number of used DoH resolvers, the number of DoH flows, and the number of Non-DoH flows.

Dataset	Sources	Servers	DoH	Non-DoH
Design dataset	*CIRA-CIC-DoHBrw-2020* [96] and First Generated Dataset [71]	12	~191 K	~1.3 M
Evaluation dataset	Collection of Datasets - Generated [65]	16	~254 K	~1.1 M
Real World dataset	Collection of Datasets - Real-World [65]	137	~1.6 M	~123 K

The whole proposal was evaluated using completely different datasets to make a trustworthy and independent evaluation. For this purpose, the generated and real-world parts from the huge collection of DoH datasets (see Section 6.2) were used. The generated part, named *Evaluation dataset*, was created similarly to *CIRA-CIC-DoHBrw-2020* and the first generated dataset; nevertheless, it was created at a different time and contained traffic towards more DoH servers; thus, its use for evaluation simulates the system's real-world deployment in the same environment. The second part represents capture from a real-world network called Real World dataset, which simulates deployment of the method in a different environment and helps evaluate the robustness and sensitivity to data drift. The summary of all used datasets is in Table 7.1[4].

7.2 The Proposed DoH Detector

The proposed detector is constrained to use traditional flow telemetry with standard features (see Section 2.2.1) only. This limitation makes the solution practical and compatible with most of the currently available monitoring infrastructure. The sole machine-learning method leads to a high number of false positives in flow classification even when achieving almost absolute accuracy considering the high throughput of a network. The proposed method, hence, does not rely solely on machine learning. Instead, the deployment of a heterogeneous detection approach was chosen. The detector utilizes three different types of detection—IP-based, ML-based, and active probing detection.

The detection pipeline uses a feed-forward loop, where a reliable but resource-consuming verification step creates a blocklist/allowlist, which is then utilized by a fast IP-based detector. To limit the number of active probes, an ML-based classifier was deployed, which selects the DoH-suspicious flows that are worth verifying. Using three different classifiers, the proposed approach overcomes each detector limitation: The obsolescence of IP lists is mitigated by active verification and continuous updates, the ML inaccuracy is mitigated by active verification, and the resources needed by active verification are minimized by IP list and ML prefiltration. The processing pipeline of the proposed detection system is depicted in Figure 7.2 labeled as DoH Detector and can be divided into six following steps:

Flow Stitching is an optional step needed to be deployed into flow monitoring infrastructure with unidirectional flow records. Since the proposed approach requires bidirectional flow records, which are still not widely adopted by flow monitoring infrastructures [53], their reconstruction from two unidirectional records is needed.

IP-based Detection uses knowledge acquired from previous IP inspections to recognize DoH resolvers. In the absence of any previous knowledge about the destination IP address, the flow is forwarded to the next step.

Filtration step performs pre-filtration of flows. When the flow is too short for reliable DoH detection, it is directly labeled as non-DoH. Other flows proceed to the following step. Further information about this step is provided in Section 7.2.4.

[4]Only flows where features can be calculated are counted hence the numbers might be different than elsewhere in work.

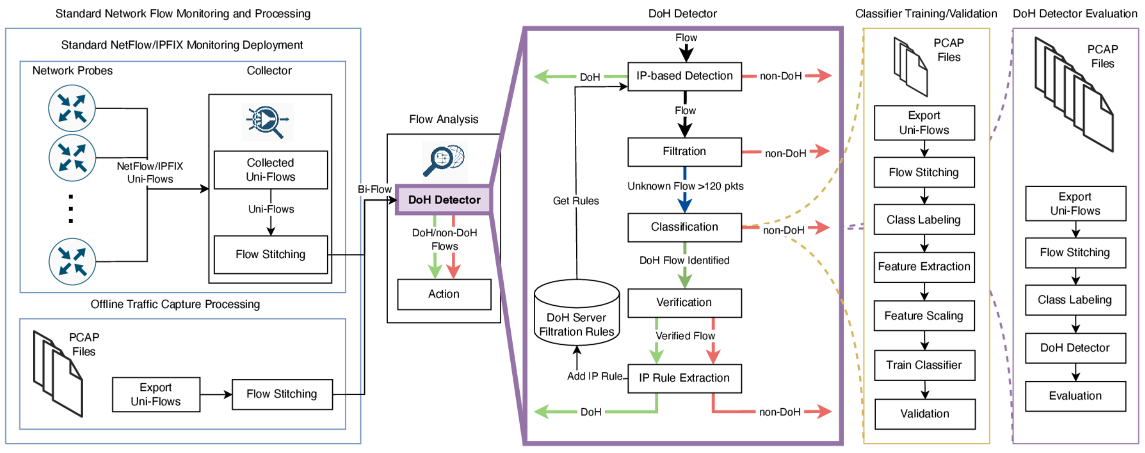


Figure 7.2: Holistic view of the proposed approach with the DoH detector design in context of the monitoring infrastructure.

Classification uses machine learning to detect DoH traffic. Moreover, this step also performs feature extraction. A detailed description of this step is provided in Section 7.2.3.
Verification step uses active DoH queries to the suspected DoH resolver to confirm its DoH resolution capability. When verified, the DoH resolver's IP address is stored in the DoH blocklist, which is then used in the filtration step. This step is further described in Section 7.2.5.
IP Rule Extraction processes the detection results and maintains a blocklist/allow list which is then used by the IP-based detector.

Each component is described in the following sections. Their description follows the order as they are depicted in Figure 7.2 except for the classification and filtration. The necessity of the filtration step arises from the limitations posed by the ML classifier and only standard NetFlow telemetry data. Hence, it is described after the classification. Measurement and demonstrations in the following sections are made on the design dataset. The evaluation and real-world datasets are held exclusively for the final evaluation of the whole system (see Section 7.3).

7.2.1 Flow Stitching

Flow stitching performs the conversion of unidirectional flow records into bidirectional. Flow stitching is a standard process that can reconstruct one bidirectional flow record from two unidirectional records. It is often placed or implemented on the collector. Multiple open-source tools are capable of flow stitching, such as BiFlow Aggregator[5] or Cisco Joy[6]. The features of bidirectional flows created by stitching are shown in Table 7.2. These flow records are then directly forwarded to the IP-based detection stage.

Naturally, the flow stitching adds some latency to the detection pipeline depending on the aggregation time window of the stitching tool. The aggregation windows should be set for particular monitoring infrastructure and its overall latency and jitter. Nevertheless, since flows from both directions are going to be exported at a similar time, the flow stitching aggregation time window could be very small in the order of seconds; hence, similar latency would be added.

[5]https://github.com/CESNET/Nemea-Modules/tree/master/biflow_aggregator
[6]https://developer.cisco.com/codeexchange/github/repo/cisco/joy

Table 7.2: Bidirectional flow record after application of flow stitching. The fields denoted in italic are statistical features and thus can be used for detection. Other features are specific to the dataset and should not be used. The table contains shortcuts for *C* and *S* which denotes client and server respectively.

Flow Record
Source IP address
Destination IP address
Source Port
Destination Port
Transport Layer protocol
Time start
Time end
Total Number of packets
Total Number of bytes
Number of packets $S \rightarrow C$
Number of bytes $S \rightarrow C$
Number of packets $C \rightarrow S$
Number of bytes $C \rightarrow S$
Duration

7.2.2 IP-Based Detection

The IP-based detection uses previously acquired knowledge about server IP addresses, maintained by IP Rule Extraction and the database, to prevent unnecessary active verification. It directly detects DoH or regular HTTPS by observing the servers' IP address field in the flow. When there is no prior knowledge about the servers' IP address, the flow is forwarded to the next stage.

7.2.3 Classification

The classification step uses a machine-learning classifier for an additional selection of DoH-suspicious flows. The description of the classification step is provided before filtration since the ML classifier showed unsatisfactory results (but comparable with previous proposals) when used with short connections and only NetFlow telemetry data. This limitation is then mitigated by the filtration step, which is described afterward. The machine-learning-based classifier relies solely on the statistical properties of the flows to learn the DoH traffic shape. This section describes the process of creating the machine learning model for DoH detection.

Feature Selection

Identifying discriminative features from the incoming flow data is one of the essential tasks during the detector design. The selected features directly influence the detector's performance. To determine features properly, the analysis of the Design dataset was performed. Moreover, the information from previous traffic observations mainly covered in Chapter 5 has been used and summarized here. The following DoH characteristics can be observed that can discriminate DoH from other HTTPS traffic:

Observation 7.2.1 *The DoH, in comparison to non-DoH HTTPS traffic,* ***transmits fewer data*** *in requests and responses.*

Observation 7.2.2 *The DoH connections last longer, containing many short transactions, creating an* ***overall higher number of smaller packets*** *in the flow.*

Observation 7.2.3 *DoH in the browser* ***creates multiple streams*** *over the same HTTP/2 connection and uses multiplexing for faster DNS resolution.*

Observation 7.2.4 *The* ***packet size variance*** *of DoH* ***is lower*** *compared to other HTTPS traffic.*

Observation 7.2.5 *The DoH connections are* ***more symmetrical****, than regular HTTPS in terms of transferred packets and bytes in each direction.*

The statistical bidirectional flow feature set is relatively limited, as seen in Table 7.2. Instead of using features directly, all possible pairs were created by applying division since it creates ratios representing potential discriminators, leaving us with 21 features—ratios of the original features. Consequently, a feature reduction step was performed by calculating the pair-wise Pearson correlation coefficient. The feature reduction removed features that showed near-perfect correlation—their Pearson correlation coefficient was higher than 0.9 [116].

Table 7.3: Selected features together with their Gini importance. The abbreviation stands for: $C \longrightarrow S$ - represents Client-to-Server direction, $S \longrightarrow C$ - represents Server-to-Client direction, *sc-bytes*/*sc-packets* represents number of transferred bytes/packets in $S \longrightarrow C$ direction, *cs-bytes*/*cs-packets* represents number of transferred bytes/packets in $C \longrightarrow S$ direction.

id	Feature Name	Gini	Formula
1	mean payload size $S \longrightarrow C$	0.293	$\frac{sc-bytes}{sc-pkts}$
2	mean time between packets $S \longrightarrow C$	0.269	$\frac{sc-pkts}{duration}$
3	mean payload size $C \longrightarrow S$	0.247	$\frac{cs-bytes}{cs-packets}$
4	num packets $C \longrightarrow S$ to packets ratio	0.191	$\frac{cs-packets}{total-packets}$

The final feature set is listed in Table 7.3 with the corresponding importance based on the computed Gini index. The final features are also related to traffic observations. The most essential feature #1 and also feature #3 capture the DoH behavior from Observation 7.2.1 and Observation 7.2.2. Feature #4 captures the flow symmetry property from Observation 7.2.5. Observation 7.2.3 is captured by feature #2. Unfortunately, used features cannot describe Observation 7.2.4 since it cannot be computed or approximated from traditional flow data.

Machine Learning Classifier

Given the aforementioned selected features, several classification algorithms were applied to the dataset. Python programming language was employed using Scikit-Learn[7] and XGBoost[8] libraries for learning classification models. Random Forest, K-Nearest Neighbors (KNN—in this case, 4NN was used, which was chosen as the best performing during the hyperparameter tuning phase), Naive Bayes classifiers, and popular boosted algorithms XGBoost and AdaBoosted Decision Trees classifiers were utilized. A detailed description of these ML-based algorithms, which are commonly used in networking tasks [15] is provided in Section 2.3.2.

The standard metrics for unbalanced datasets were used as described in Section 2.3.3 to determine the best-performing algorithm for DoH detection. In particular, *precision* (2.15), *recall* (2.16), and

[7]https://scikit-learn.org
[8]https://xgboost.ai

F1 measure (2.17) were calculated[9]. The primary measure is the *F1 measure*, showing the classifier's overall performance. However, the precision metric is also important since practical applications in network monitoring aim to reduce the number of false positives.

The process of algorithm evaluation followed a standard machine-learning classifier design. First, the Design dataset (see Table 7.1) was divided into the train (75%) and test (25%) parts using stratified sampling. The train part was used for hyperparameter tuning using cross-validation and training the classifier, and the validation part was reserved for classifier evaluation.

Then, the features within each part were standardized using Z-score normalization (also called standardization) since the feature values fall into different ranges. Feature normalization was applied, particularly for features #1, #2, and #3 from Table 7.3, which contain values ranging from 0 to hundreds.

Table 7.4: Performance metrics of classifiers.

Algorithm	**F1**	**Precision**	**Recall**
XGBoost	**0.973**	**0.981**	**0.966**
Ada-boosted DT	0.970	0.978	0.963
KNN	0.963	0.972	0.955
Random Forest	0.955	0.979	0.931
Naive Bayes	0.422	0.304	0.693

Each algorithm requires setting proper input parameters, also called hyperparameters. The hyperparameter tuning of all algorithms was done experimentally using a grid-search technique and 5-fold cross-validation on the train part. Models with the best hyperparameters were then trained on the whole train part and evaluated on the test part. Table 7.4 presents performance results computed using standard metrics numerically. Most algorithms performed well, capable of achieving an F1 score of higher than 0.95, except for the Naive Bayes. Among them, the best performance is achieved by boosting algorithms. The XGBoost algorithm had the best performance and achieved $F1 = 0.973$ with the following hyperparameters: maximum depth of seven and subsample of 0.9.

Table 7.5: Confusion matrix for XGBoost classifier.

		Predicted	
		DoH	**HTTPS**
Actual	**DoH**	46 192	1643
	HTTPS	881	281 609

The detailed performance of the best-performing XGBoost classifier for DoH recognition can be seen in Table 7.5. Although the F1 of 0.973 can be considered high accuracy, it may still not be accurate enough for deployment on the computer network due to a high number of predictions per second. According to Hofstede et al. [52], the Czech national research and educational network creates up to 10 000 flows per second—the same number of predictions per second would be made by the classifier when deployed on this network. Therefore, the classifier would produce around 2.6 false positive[10] detections per second (9360 false positive detections per hour). Generating such

[9]Abreviations have following meaning: True Positives (TP), False Positives (FP), False Negatives (FN), True Negatives (TN)

[10]FP/(FP+FN+TP+TN) = 0.0026

a large number of misclassifications would overwhelm the administrators and the system, causing such detection not to be beneficial. Hence, there is a need to push the detection system toward more accurate detection.

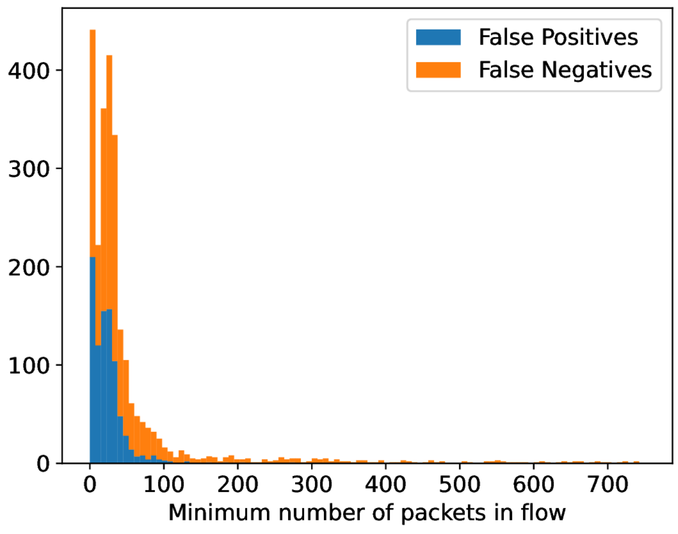


Figure 7.3: Stacked histogram of misclassified flows.

Further investigation revealed that short flows are the main cause of reduced accuracy. Figure 7.3 shows the number of false positive detections depending on the number of packets in flows. It can be seen that misclassification happens more often when flows have less than 100 packets. To reduce false positives, the elimination of shorter flows is needed since those cannot be reliably recognized by the machine-learning classifier that uses standard flow information.

7.2.4 Filtration

The DoH connection (similarly to other HTTPS connections) involves a TCP handshake, Transport Layer Security (TLS) handshake, HTTP/2 preface, application data transfer phase, and TCP termination, as shown in Figure 7.4. Each phase amounts to several packet exchanges. Based on the measurement of single query DoH presented in Section 5.1, it was determined that one DNS request and response over HTTP/2 requires (on average) 28 packets (HTTP/1.1 requires an average of only 17 packets[11]), with only two packets carrying the actual DNS payload. The number of packets in the HTTP/2 preface can be even higher since the single query measurement does not need to set up multiple channels as other more complicated connections made by browsers might do.

The shorter flows might not contain any DoH queries at all. Browsers usually create multiple connections to the target server to optimize the loading process. The browser contacts the server with several connections—at least one connection is then used, and the rest are failed or terminated[12]. Such connections do not transmit any application data.

Hynek et al. [62] mentioned short DoH connections as challenging. Particularly, the single-query DoH produces flows with similar statistics as other short API calls. The influence of many HTTP/2 preface packets compared to the small portion of DoH data is high and complicates reliable

[11]Lower number of packets is caused by missing novel HTTP/2 features such as connection preface, multiplexing and HTTP header and payload transmission separation.

[12]The number of parallel connections created by a browser is not specific for DoH and can often be configured [122].

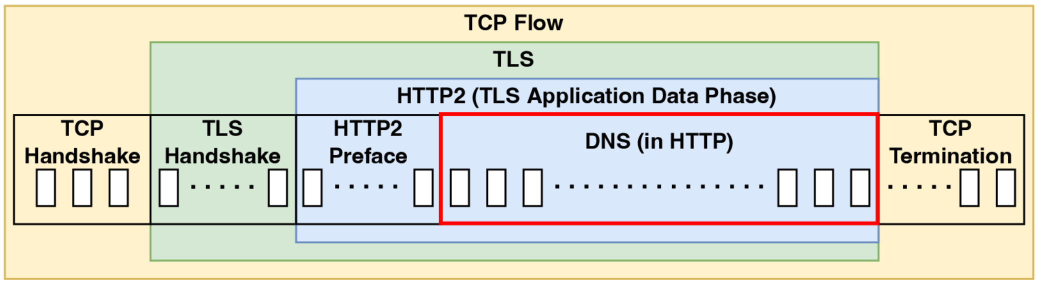


Figure 7.4: Schema of HTTPS (DoH) flow with all connection prefaces highlighted.

DoH detection. Therefore, the determination of the minimal number of packets in the flow carrying enough DoH packets is needed to recognize DoH from other HTTPS traffic reliably.

Determining the Minimal Number of Packets Threshold

There are multiple ways to determine the minimal number of packets in flow needed for reliably distinguishing DoH from regular HTTPS traffic. One of the approaches would be to add the threshold to hyperparameters and find the best value during the hyperparameter tuning phase. Another way would be to use an unsupervised clustering method to group samples automatically and measure the ability to separate samples into two classes by measuring the intra-class similarity. Both approaches were used.

At first, the experiments with threshold tuning were performed. The best-performing XGBoost algorithm was used, and the minimum packet threshold was incrementally increased. The model was retrained each time the threshold was raised. As expected, the model's accuracy increases with a higher threshold, as shown in Figure 7.5. The model reaches first stable performance around 116 packets and then maintains similar accuracy.

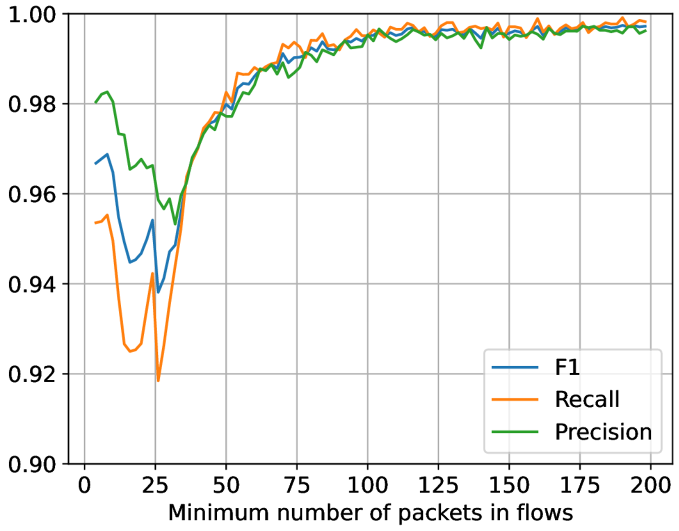


Figure 7.5: Minimum number of packets in flow hyperparameter tuning.

In the second approach, the K-Means [47] clustering method was used with k-means++ initialization to determine the minimal packet lengths. It was done by using the algorithm's ability to group samples into a predefined number of k clusters (in this case, $k = 2$, the same as the number of classes) based on their similarity. Applying the algorithm to all flows without filtration would lead to a different capability of distinguishing the samples reliably than when applied to filtered flows.

The dataset without filtration was clustered, and then the required number of packets in flows was increased. The intra-class similarity was measured by calculating the purity score of the resulting clusters. Since the dataset is highly imbalanced in favor of the non-DoH class, random undersampling was applied before each clustering to balance the dataset and correctly compute the purity score. Undersampling rather than over-sampling was chosen since there was enough data to work with, and it works only with real data records without the need for any data augmentation.

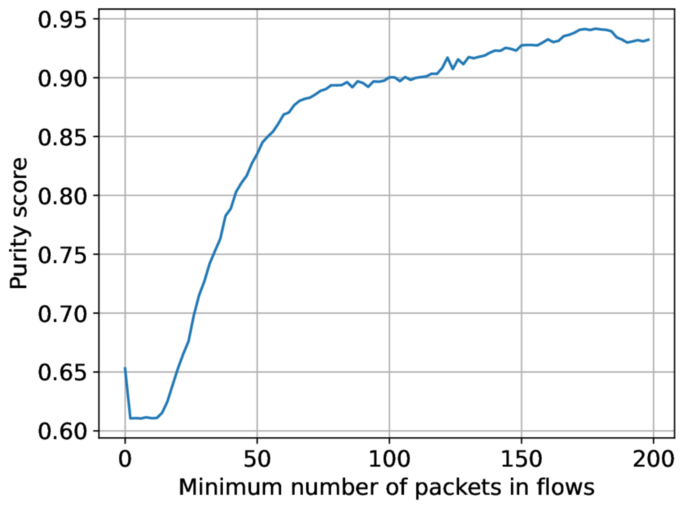


Figure 7.6: Purity score of the converged clusters based on minimum packets in flows.

The results are depicted in Figure 7.6. It can be seen that the minimal number of packets in flow highly influences the DoH recognition capability of the clustering method. More packets in flows make the recognition more reliable, which is consistent with previous observations. The 90% purity index of both clusters is reached when flows have at least 112 packets.

In addition, the estimation of the average number of packets in the DoH connection needed for fetching one website was performed. The DoH connections of the top 10 000 visited websites from the *Majestic Million* dataset were investigated. The average number of unique domain references by these websites was 20, leading to an equal number of DNS resolutions. Measured in terms of packets, this gives roughly 120 packets in a DoH flow corresponding to one website visit.

All experiments showed satisfactory DoH detection performance with similar values ranging from 112-116 packets, while flows with more packets are generally more reliably recognized. It was decided to set the filtration threshold to ≥120 packets since such a number of DoH packets is generated during an average single web page visit. According to Crichton et al. study [30], users mostly visit more than one site in a short period and spend on average 1.7 hours browsing in 35.9 browser tabs daily. Even when considering DNS caching mechanisms, there is a high probability of generating longer DoH flows than the estimated 120 packets. In practice, the threshold thus could be set to even higher numbers of packets in a flow, which would improve the model accuracy even more.

Final Classifier Evaluation After Filtration

The final experiments of the detection module accuracy were performed on the flows that fulfilled the requirement for the minimal number of packets—the flows with less than 120 packets were filtered out. The filtration reduced the number of DoH flows to 16 728 (8.74% of the original dataset) and 123 222 (9.34%) non-DoH flows. Despite a more than 90% reduction in flows, the number of unique DoH servers and their IPs remained the same.

Table 7.6: Performance metrics of classifiers on reduced dataset—it contains only flows with ≥120 packets. (See Table 7.4 for comparison.)

Algorithm	**F1**	**Precision**	**Recall**
XGBoost	**0.996**	**0.995**	**0.997**
Random Forest	0.995	0.993	0.997
Ada-boosted DT	0.994	0.991	0.996
KNN	0.993	0.991	0.996
Naive Bayes	0.927	0.887	0.971

The accuracy of classifiers reached on the reduced dataset is shown in Table 7.6. The classification methodology remained unchanged, with the data split into the train (75%) and the test part (25%) using stratified sampling. Nevertheless, hyperparameter tuning was not performed, and the same hyperparameters as in the previous evaluation (described in Section 7.2.3) were used. The best performing algorithm was again the XGBoost, achieving F1 of 0.996.

Table 7.7: Confusion matrix for XGBoost classifier evaluated on the reduced dataset—it contains only flows with ≥120 packets.

		Predicted	
		DoH	**HTTPS**
Actual	**DoH**	4169	13
	HTTPS	20	30 786

The detailed results of the XGBoost algorithm are provided in Table 7.7. It can be seen that the number of false positives is significantly reduced (see Table 7.5). Since the number of misclassifications is relatively low, they could be investigated manually. The 20 false positives were generated mainly by telemetry and analytical services embedded in the websites by advertisement. These types of flows show similar traffic shape characteristics as DoH, which justifies their misclassifications. The false negatives can be identified as anomalous, with at least one feature value standing out or at the edge of the decision boundary.

7.2.5 Verification

The next step in the DoH detection pipeline is active server verification. Active verification is the most accurate way used in the past by multiple researchers [130, 136].

The active probing module generates DoH requests, for *example.com* domain to */dns-query* endpoint via both HTTP GET and HTTP POST methods, as specified in [51]. The servers hosted on shared infrastructures usually require a domain name for a successful connection. Therefore, the active verification can use a domain name from TLS handshake when extracted by monitoring appliances, available passive DNS services (providing IP-domain mappings), or direct IP queries. Passive DNS and direct IP queries are the only options in this case where the lack of extracted domain name is present. Moreover, additional paths such as root path may be tested if the RFC defined path is not responding.

The active probing approach always comes with ethical consideration and the correct setting of the probing rate. When using an aggressive rate, the source IP address could be marked as malicious and blocked by service providers, resulting in reduced verification efficiency. Moreover, active requests always consume some target resources. Nevertheless, in this case, the HTTPS servers are probed; thus, they should be scaled adequately for HTTP requests. Moreover, by storing even non-DoH server IP addresses, it was ensured that the active probing module checks each IP address/domain only once in 24 hours, thus limiting the target resource utilization to a minimum.

7.2.6 IP Rule Extraction

The verification results are then processed, and the servers' IP addresses are extracted and stored in the database IP DoH Server Filtration Rules, as seen in Figure 7.2. Both results, the confirmed DoH and confirmed non-DoH servers, are stored in the database and used for direct detection in the IP-based detection step. The database then limits the amount of ML-based detection and verification, thus increasing the whole system's performance.

As Gracia et al. [44] discussed, the detection of DoH by blocklists is inefficient due to fast blocklist obsolescence. Consequently, the non-DoH addresses are stored only for 24 hours, and the DoH resolvers are regularly checked (every 24 hours) to maintain the timeliness of the stored information.

7.3 Experimental Results

The whole proposed DoH detector depicted in Figure 7.2 was evaluated using a separate Evaluation dataset (see Section 7.1). Using an utterly separate dataset captured in the same network environment with different properties not seen during the design phase is similar to actual deployment, with the benefits of the ground truth labels. Thus, the correctness of the evaluation performance can be guaranteed.

The evaluation simulated the deployment of the system in the same network environment. The dataset PCAP files were processed with the NetExP tool (as in the case of the Design dataset). Since the NetExP creates bidirectional flow records, its extraction also represents a flow stitching phase. The flows were processed in the original order by the detection pipeline. The XGBoost classifier was used as proposed in Section 7.2.4, which was trained on the train part of the Design dataset. Since the Evaluation dataset is more than one year old, active verification was not performed; instead, the ground truth labels provided within the dataset were used. The evaluation process was set to classify one flow at a time without any parallelization and other performance optimization techniques.

The final evaluation dataset contained 254 788 DoH flows, and 1 142 329 non-DoH flows. Nevertheless, there are only 25 078 DoH, and 137 327 non-DoH flows with more than 120 packets that would pass the filtration step. The confusion matrix of the overall system evaluation is shown in Table 7.8. The system achieved an F1 of 0.998 with a precision of 1.0 and a recall of 0.995. No false positives have been observed, which is expected due to the verification stage. However, there are 1147 false-negative flows.

Despite the false negatives, the system was capable of identifying all DoH servers' IP addresses in the dataset—thus, the created blocklist was complete. All false negatives occurred due to the latency of the DoH server identification. From 16 different resolvers, 15 were selected by ML for verification during the first occurrence of flow with more than 120 packets; the missing one was selected after several occurrences. However, almost all false negatives (except one) were then created by the filtration module. Before the first flow with more than 120 packets, the filtration module automatically marked shorter connections as non-DoH. On average, the system falsely marked nine short DoH flows before a longer DoH flow was received and successfully identified as DoH.

Table 7.8: Confusion matrix representing the final evaluation of the system.

		Predicted	
		DoH	**HTTPS**
Actual	**DoH**	252 641	1147
	HTTPS	0	1 142 329

Overall, the dataset contained flows toward 76 870 unique IP addresses. The system labeled 23 unique DoH server IP addresses (some of the DoH resolvers had more than one address) and allowlisted another 87 IP addresses as false positives. It is worth noting that those false positives created by ML are expected and handled by the Verification and IP-based Detection stages. Thus, the whole detection system will not raise false alarms. In total, only 110 flows were forwarded to the verification stage. Assuming a large network with 10 000 flows per second, the Evaluation dataset represents 139.7 seconds of traffic, meaning the active verification would create ~0.7 active requests per second, which can be considered very low and acceptable for real-world deployment.

7.3.1 Evaluation of the Data Drift

The network traffic is often susceptible to data drift [88] (sometimes called concept drift)—a phenomenon in which the underlying distribution of the data changes in time due to novel services or updates of the network infrastructure. Thus, the robustness of the novel method to data drift was evaluated. The Real World traffic captured in the real backbone network [65] was used for that. This traffic was captured a year apart from the design data on an entirely different network setup. We used the proposed detector with ML trained on the design dataset and evaluated it on the Real World traffic dataset with the same methodology as in the previous section. The performance results are shown in Table 7.9.

Table 7.9: Confusion matrix for data drift susceptibility of the system.

		Predicted	
		DoH	**HTTPS**
Actual	**DoH**	1 549 380	42 210
	HTTPS	0	123 050

The proposed detector achieved an F1 score of 0.987 and an accuracy of 97.5%. Given the different training and testing network environments and the year gap between training data and testing data, which might be considered an extreme case, the detector showed stable performance with only ~ 0.01 F1 score drop and a 2.4 percentage point drop in accuracy. For comparison, other work also dealing with encrypted network traffic and data drift of Malekghaini et al. [88] experienced a drop of up to 40 percentage points when trained and evaluated on year-apart data.

Table 7.10: Comparison of related work metrics. The abbreviations stands for: **NF** - Number of required statistical features; **D** - Dataset; **S** - Number of DoH Servers in the dataset; **SFE** - Short flow elimination before passed to ML model; **DoHBrw** - *CIRA-CIC-DoHBrw-2020* dataset [96]; pkts - Packets; ppkts - Payload packets.

Paper	S	NF	Features	Dataset	SFE	Best algorithm	F1	Accuracy
Proposed method	**12**	**4**	**Standard**	**DoHBrw, Custom**	**< 120 pkts**	**XGBoost**	0.998	99.9%
Vekshin et al. [129]	2	18	Extended	Custom	< 5 ppkts	Ada-Boosted DT	-	99.6%
MontazeriShatoori et al. [97]	4	28	Extended	DoHBrw	-	Random Forest	0.993	-
Banadaki[a][10]	4	34	Extended	DoHBrw	-	LGBM	-	100%
Jha et al. [72]	2	N/A[c]	Extended	Custom	-	DeepFM	0.995	-
Casanova et al. [21]	4	28	Extended	DoHBrw	-	BiLSTM	0.987[b]	99.0%
Behnke et al. [12]	4	26	Extended	DoHBrw	-	Random Forest	0.998	-
Mitsuhashi et al. [94]	4	28	Extended	DoHBrw	-	XGBoost	0.998	99.8%
Nguyen et al. [105]	4	29	Extended	Custom	-	Transformer NN	0.99	-
Konopa et al. [75]	N/A[c]	3	Standard	Custom	-	Feed Forward NN	-	94.4%
Zebin et al. [138]	4	29	Extended	DoHBrw	-	Balanced Stacked RF	0.999	99.98%

[a] There is concern about general applicability of this detector, since the classifier is fitted on IP addresses from the used dataset.
[b] Score is computed from provided confusion matrix.
[c] Authors do not provide this information.

7.4 Discussion

The proposed system performs similarly or better than the other related works as summarized in Table 7.10, while trained and validated on a more comprehensive DoH dataset containing traffic towards more DoH resolvers including real-world traffic where different resolvers and their configuration highly influences the DoH traffic shape (as shown in Chapter 5). Due to the rich dataset and thorough analysis, this is the first work that points out the short DoH flow phenomena and designs the detection pipeline accordingly. Moreover, the majority of previous works suffer from reliance on specialized hard-to-obtain features. The tailored features improve the accuracy, but they also limit mass deployment due to the necessity of installing specialized network probes.

This proposal works with easy-to-obtain features available in almost all network monitoring infrastructures, even those with NetFlowV5. The only related work that also works with features extractable from traditional NetFlow is the Konopa et al. [75], who achieved an accuracy of 94.4% in DoH detection. Compared to them, the proposed system achieved 99.9% accuracy while evaluated on the biggest DoH dataset available. The proposed system also addresses the desirable elimination of false positives. The increased precision is due to the use of three heterogeneous classifiers, each of which has advantages and limitations that together create a more robust and precise DoH detector.

While the goal was to minimize false positives, the detector can still produce some when DoH and non-DoH services are both hosted behind a single IP address. Such cases cannot be distinguished by IP addresses but by domain names. Since the majority of flow monitoring infrastructures support only NetFlowV5 [53] and do not support the extraction of domain names from TLS handshakes, it might be considered a limitation, even though it is caused by the underlying network monitoring infrastructure. When extraction of domain names is supported, modification of the presented architecture is trivial (block list would use domain names instead of IP addresses), and the false positives would be mitigated.

The detection system also suffers from false negatives, mainly caused by the detection latency. Nonetheless, the majority of false negatives could be mitigated by back-labeling of flows automatically labeled as non-DoH by the filter. It would require additional temporal storage of short flows that would wait for at least several seconds for the potential labels because a long one often follows short DoH flows (as discussed in Section 7.2.4).

Furthermore, skilled adversaries can bypass detection by limiting the number of packets in their DoH connections. Nevertheless, such packet limitation would significantly impact the DoH performance. Moreover, DoH only encrypts DNS, and more skilled adversaries would rather use other stealthy communication, such as HTTPS-based VPN or other multipurpose tunneling tools.

Despite all the mentioned limitations (mainly caused by the limited availability of traffic features), the proposed DoH detector proved to be accurate and computationally efficient. Most work is done using simple list-based filtering and classification using a pre-computed model. Only a fraction of inputs require verification through active probing. Moreover, the detector's input consists of standard NetFlow records available on various network devices.

Table 7.10 provides a supporting summary of the related works compared to the proposed method and shows the data source and other characteristics in which the methods were created and evaluated. However, the performance of all methods can not be qualitatively compared since not all methods used the same dataset. Hence, the final performance of the methods may differ. Methods should be compared in similar conditions while using the same data and multiple scenarios. Such a comparison and testing is needed and provided in the following chapter.

Chapter 8

Comparison with Existing DoH Detection Approaches

Over the past years, there have been multiple ML-based proposals for DoH detection, often surpassing the 99% detection accuracy. These novel ML-based DoH detectors have often been evaluated using a different lab-created dataset from a limited time period. However, without using the same data, the proposals cannot be qualitatively compared; thus, we still lack the answers to basic questions such as „*What is the most accurate detection approach?*" or „*How do the detectors behave in long-term?*". Therefore, this chapter aims to compare several ML-based DoH detection approaches with the proposed one in six data scenarios, answering six different research questions[1].

Methodologies of the published works for DoH detection were followed, and detectors were recreated to evaluate their properties using a previously presented Collection of datasets with DoH traffic (see Section 6.2). Additionally, subsequent capture of real-world DoH traffic as described in Section 6.3 was used to study the long-term properties of ML-based DoH detectors. These datasets allowed to evaluate DoH detectors thoroughly and finally show the advantages and disadvantages of published DoH detection approaches and the proposed one.

8.1 Used Datasets

For the comparative analysis, the comprehensive collection of datasets described in Section 6.2 was used, which is provided in the form of PCAP files, allowing extraction of all features used by the detector proposals.

Together, the collection of datasets provides a comprehensive set of DoH and regular HTTPS data. The real-world part contains timing and other network characteristics as present in a real network, while the generated part contains a large set of DoH resolvers, each with slightly different characteristics. The two datasets are very different in terms of packet timing and size distribution. They are thus ideal for assessing the usability of models trained on lab-created data in the real-world environment.

Nevertheless, the collection of datasets does not allow for experiments with data drift—a phenomenon where the distribution of the input data changes over time, which causes the obsolescence of trained ML models. There have been numerous reports about ongoing data drift [88]. To allow experiments concerning the longitudinal usability of the detectors, the additional 5-week capture described in Section 6.3 was used. The information about all used types of datasets is provided

[1]The results presented in this chapter were also applied for publication by the author at the time of finishing this thesis.

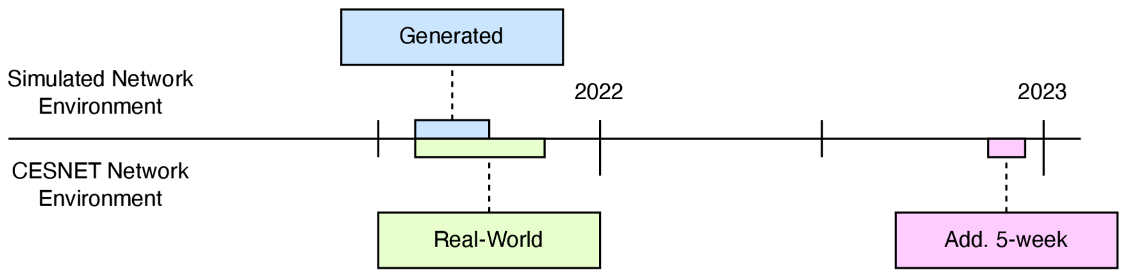


Figure 8.1: Timeline of dataset capturing.

in Table 8.1, and the datasets involved in the comparison together with the times of dataset capturing are graphically shown in Figure 8.1.

Table 8.1: Statistics about used types of dataset. Please note that the number of connections is not equal to the number of flows referred elsewhere in the work since each flow exporter performs different connections splitting into multiple flows with an active/inactive timeout mechanism.

Type Name	**Other Connections**	**DoH Connections**	**Size**
Real-World	~156 K	~5.2 M	179 GB
Generated	~1.6 M	~346 K	250 GB
Add. 5-week	~17 K	~486 K	52 GB

8.2 Selected DoH Detection Approaches

Several machine-learning approaches were published to distinguish DoH and regular HTTPS (non-DoH). Some of them focus only on the DoH detection task, while others include the task as the first stage, and in the second stage, they further focus on malicious DoH detection. Alternatively, some approaches focus solely on malicious DoH detection. Since the method proposed in the in Chapter 7 (further denoted as *proposed method*) focuses on the detection of DoH in regular HTTPS and works with flows, the comparison is limited to DoH recognition and flow methods only.

Table 8.2: Criteria summary for comparison of published DoH detection methods.

Paper	**PCAP processing**	**Features**	**No identifiers**	**Defined architecture**
MontazeriShatoori et al. [97]	✓	✓	✓	✓
Mitsuhashi et al. [94]	✓	✓	✓	✓
Behnke et al. [12]	✓	✓	✓	✓
Casanova et al. [21]	✓	✓	✓	✓
Zebin et al. [138]	✓	✓	✓	✓
Vekshin et al. [129]	✓	✓	✓	✓
Konopa et al. [75]	✗	✓	✓	✗
Jha et al. [72]	✓	✗	✓	✓
Banadaki [10]	✓	✓	✗	✓
Nguyen et al. [105]	✓	✓	✓	✗

Out of all published approaches, only those methods that satisfied the following criteria that enable fair comparison and reproduction of the proposed methodology were chosen to be compared along the *proposed method*:

1. published method of PCAP processing and feature extraction;

2. precisely defined features that were used for detection;
3. not used identifiers (IP addresses, ports) as features;
4. defined architecture of the algorithm used for classification (e.g., number of hidden layers).

As shown in Table 8.2, only the studies of MontazeriShatoori et al. [97], Mitsuhashi et al. [94], Behnke et al. [12], Casanova et al. [21], Zebin et al. [138], Vekshin et al. [129] satisfied the conditions. Some works did not include a tool or description of flow extraction from PCAP files and could not be reproduced; other studies did not provide the exact features they involved in the detection. Furthermore, the works that used flow identifiers such as IP addresses or ports in the feature vector were not included. As discussed by Behnke et al. [12], these features would overfit the lab-created datasets, which prevents their comparability with other approaches.

The selected approaches and the *proposed method* used different methodologies that included traffic processing tools, flow elimination, balancing, scaling, and a number of features and achieved their best results with different algorithms. Table 8.3 summarizes the selected approaches with the mentioned metrics.

Table 8.3: Selected DoH detection approach summary. The abbreviations stand for: **NF** - Number of required statistical features; **FE** - Flow elimination before passed to ML model; pkts - Packets; ppkts - Payload packets.

Paper	**NF**	**Tool**	**FE**	**Balancing**	**Scaling**	**Best algorithm**	**F1**	**Accuracy**
MontazeriShatoori et al. [97]	28	DoHLyzer	NaN	-	-	Random Forest	0.993	-
Mitsuhashi et al. [94]	28	DoHLyzer	NaN	-	-	XGBoost	0.998	99.8%
Behnke et al. [12]	26	DoHLyzer	NaN	-	-	Random Forest	0.998	-
Casanova et al. [21]	28	DoHLyzer	NaN	resampling	min-max	BiLSTM	0.987[a]	99.0%
Zebin et al. [138]	29	DoHLyzer	NaN	SMOTE	min-max	Bal. Stacked RF	0.999	99.98%
Vekshin et al. [129]	18	ipfixprobe	NaN, < 5ppkts	SMOTE	-	Ada-Boosted DT	0.976[a]	99.6%
Proposed method	4	NetExP	NaN, < 120pkts	-	standard	XGBoost	0.998	99.9%

[a] Score is computed from provided confusion matrix.

8.3 General Methodology

Six selected approaches and the *proposed method* differ in raw data processing, extracted features, algorithms used, and accompanied preprocessing. The common methodology was established to treat all the subjected approaches the same way for all prepared cases. General processing methodology is depicted in Figure 8.2. For the implementation, an automated pipeline was created that ensured no deviance in the methodology for each evaluated detection proposal to avoid any accidental mistakes and maintain comparable results.

8.3.1 Data Preparation

At first, the datasets described in Section 8.1 are in the form of raw PCAP files. There are three groups, the Real-World, Generated, and Add. 5-week, each consisting of several PCAP files. The first two groups of files are treated differently than the third. Each file in the first two groups is split in a ratio of 70:30 such that new files dedicated to train and holdout parts are created. The train part is dedicated to hyper-parameters tuning of the algorithms, and then it is used to train the model for

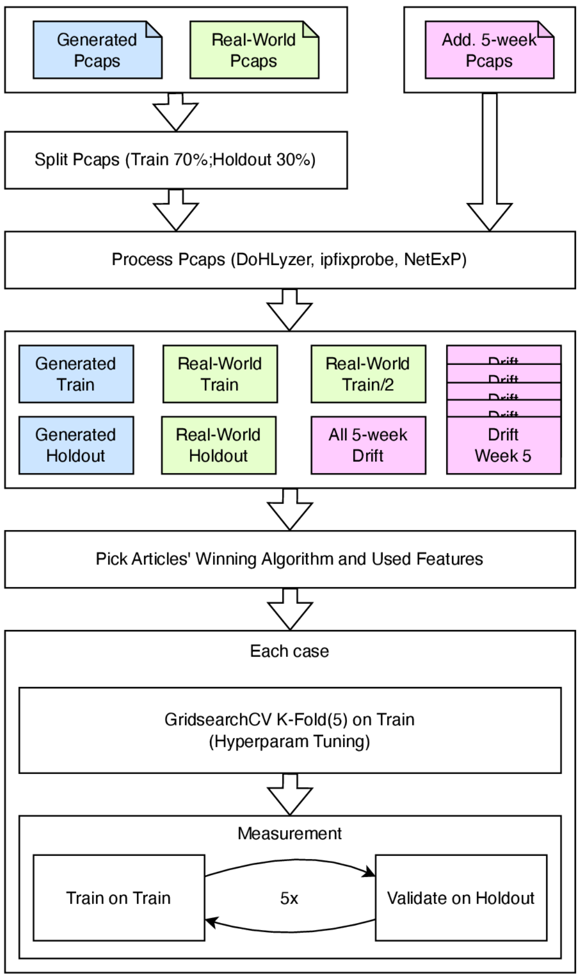


Figure 8.2: Structure of general processing methodology common for all cases.

performance evaluation. The holdout part is kept exclusively for final performance evaluation. The splitting follows the machine learning best practices also used by other works.

All the PCAP files are processed by each processing tool used in the selected studies (namely DoHLyzer, NetExP, and ipfixprobe) in the same way as in the original experiments. The data are split in its raw PCAP format before processing because each tool process flows differently with varying timeouts generating a nonidentical number of flows on the output. This way, it can be guaranteed that the original source is the same as the input, and the tool's processing is considered part of the study approach. Processed PCAP files are merged into several datasets that later serve in different combinations as input into different cases for answering research questions (see Figure 8.3).

8.3.2 Algorithm Selection and Tuning

Only the algorithms that achieved the best classification performance in the concerned studies were chosen to represent each approach (see Table 8.3). The algorithms are fed with the appropriate features. Moreover, scaling and balancing were used, as the authors described in their works. The algorithms were tuned using grid-search and K-Fold (5) on the train part before each measurement. The Bi-directional LSTM architecture was left intact; only hyper-parameters, such as the learning rate or optimizer algorithm that the authors also tuned, as they mentioned in their work [21], were tuned. The best hyper-parameters were found for each training dataset (see training parts of each research question in Figure 8.3). Each study model's hyper-parameters were tuned for each training dataset separately since each training dataset has different characteristics and sizes. That way, it can be achieved fair comparison.

8.3.3 Measurement

For the final measurement, each model was trained on the whole train part using the best-found hyper-parameters for each training dataset, and then the performance was measured on the holdout part of each belonging training dataset. All the seeds were left intact (not fixed) and randomized by default at the measurement part, and for each of the five repeats, the model was retrained before each measurement. Instead of taking a score of only one run that can represent a best or worst achieved score based on cherry-picked seeds, rather the mean and deviation upon several runs with random seeds are presented. This way, more realistic results can be shown, including the stability of the algorithm results and the boundaries between which the performance can fall in reality.

Table 8.4: The percentage of DoH flows that were filtered by each flow exportation tool due to the impossibility of feature computation. This was usually caused by a low number of packets and unidirectional communication. Such filtering results in a false-negative classification of the filtered DoH flows.

Holdout Dataset	**DoHLyzer**	**ipfixprobe**	**NetExP All**	**NetExP Filter**
Generated	24.3%	26.4%	32.8%	93.6%
Real-World	28.0%	50.9%	17.4%	97.1%
Add. 5-week	26.6%	26.3%	38.7%	95.8%

Since imbalanced datasets are assessed, to measure the performance, an F1 score was chosen, which is a recommended measure when dealing with imbalanced data. However, all methods have different constraints on the input flows. For example, Vekshin et al. [129] require flows with at least five packets; shorter flows are simply removed from the classification task, and the performance metrics are reported without them. Similarly, Montazerishatoori et al. [97] Mitshuhashi et al., Behnke et al. [12], Zebin et al. [138], and Casanova et al. [21] discards flow with *NaN* values[2] and reports all the performance metric without considering them. The percentage of filtered DoH flows in each dataset is shown in Table 8.4. It can be observed that the ipfixprobe tool used by Vekshin et al. [129] discards more than 50% of all DoH flows in the Real-World dataset and directly classify them as non-DoH. The *proposed method* using the NetExP tool filters *NaN* values (denoted as All) similarly to other methods using DoHLyzer and flows with less than 120 packets (denoted as Filter), including those with *NaN* values; hence, the drop is high, reaching up to $\sim$ 97% in the case of Real-World dataset. Not considering filtered flows in the performance metric does not reflect the true performance of the classifiers since a lot of DoH connections could be missed by such prefiltration. Therefore, two distinctive metrics are used:

F1-classified is computed just from the flows that were admitted to the classification. This metric was mainly used across the detection proposal studies.

[2]They occur due to the impossibility of feature computation—usually division by zero or short packet sequence.

F1-all is computed from all flows in the dataset. The flows that were not admitted to the classification due to the filtration are assigned with a non-DoH prediction label—similarly, as it would be in the real deployment.

Measurement of the Proposed Method

All selected studies in Section 8.2 presented only a single DoH detector proposal except the *proposed method*, where three main classifier parts are presented. The 1) ML-based classification of all flows (further denoted as *All Flow scenario*), the 2) ML-based classification of flows that contain more than 120 packets (further denoted as *Filtered scenario*) for improving accuracy, and the 3) approach utilizes the ML model from the Filtered scenario for the creation of a DoH resolver blocklist. This blocklist is then used for classification, even very short flows (this approach is further denoted as a *Simulation scenario*).

Since the *proposed method* presents the results of the three parts separately, three results are also compared. The All-flows and Filtered scenarios are directly comparable with all other approaches since they utilize ML-only. Nevertheless, the simulation scenario utilizes active probing and blocklist, giving the classifier a significant advantage in some test cases, which should be considered when interpreting the results.

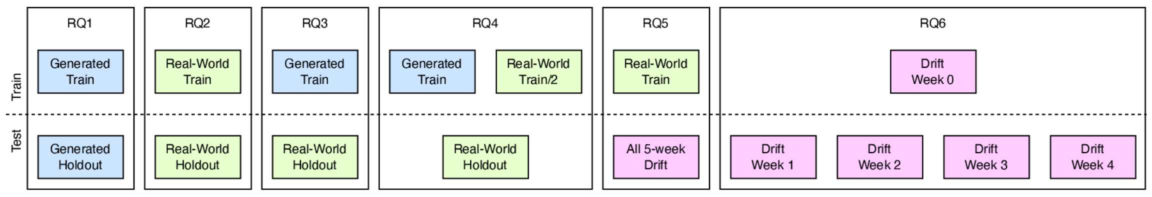


Figure 8.3: Processed datasets that are used as input to answer defined research questions.

8.4 Comparative Analysis

The analysis is divided into six research questions (RQ) and thus also experiments. For each experiment, the methodology described in the previous Section 8.3 was followed. The created automated pipeline ensured no deviance in the methodology for each evaluated detection proposal to avoid any accidental mistakes and maintain comparable results. The results are presented in the following sections.

8.4.1 RQ1: How effective is the proposed method compared to other DoH detection approaches on a lab-created dataset?

This research question concerns the general reproducibility of the selected DoH detection proposals and the *proposed method* since all of them have been primarily evaluated using lab-created data. As demonstrated in Figure 8.3, the pre-split Generated Train part was used to train the models and then the Generated Holdout to test their performance.

Results

The results of the experiment are shown in the Figure 8.4. The experiment confirmed the reported performance of each proposal, and the excellent accuracy of the methods was achieved, often reaching up to a mean F1-classified score of 0.99. The only exception that did not reach the expected

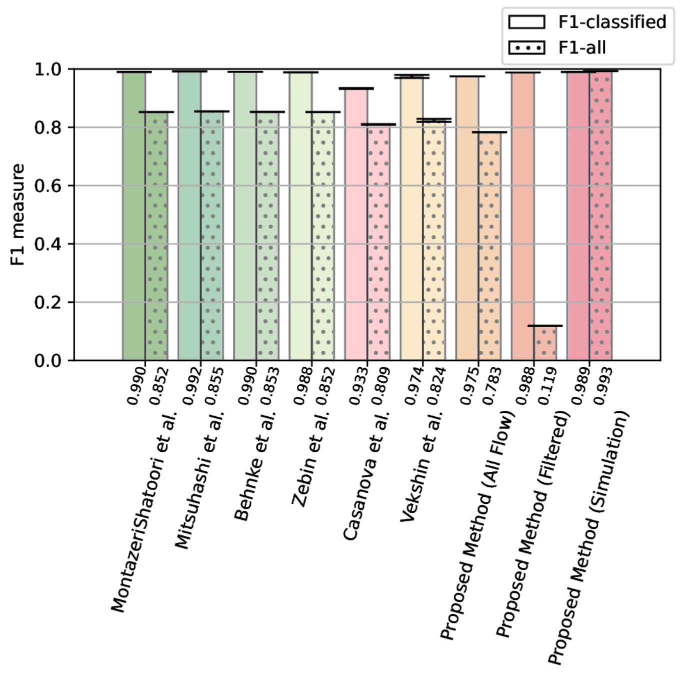


Figure 8.4: RQ1: The performance of DoH detection proposals trained on the generated dataset and tested on the generated dataset. The exact measured values of both F1 scores are written under each column. The whiskers in the plot represent the standard deviation observed during different random seeds.

accuracy was the Casanova et al. [21] who reported in their study an F1 score of 0.987. Despite the increased effort spent in hyperparameter tuning and special care in model recreation, a slightly lower F1 score of 0.93 with a very low standard deviation of 0.001 across the runs was achieved.

As expected, the measured F1-all performance shows a significant decrease compared to F1-classified. The highest accuracy drop between the F1-all and F1-classified experienced the *proposed method* Filtered, which discards flows shorter than 120 packets. Then followed by Vekshin et al. [129], who discard flow shorter than five payload packets. On the other hand, the *proposed method* Simulated showed the highest F1 score on F1-all where only a few flows were missed before correctly classifying all other flows.

Answer RQ1: Almost all DoH detection approaches performed similarly well. There is no significant difference in the results. Even the all-flow scenario of the *proposed method* shows comparable performance to the other methods while working with only four features. However, the results are slightly lower together with Casanova et al. [21] and Vekshin et al. [129].

When looking at the F1-all measure, *proposed method* Simulated can be considered the most effective method, followed by other methods that use DoHLyzer as the datasource, namely MontazeriShatoori [97], Mitsuhashi et al. [94], Behnke et al. [12] and Zebin et al. [138] since they perform minimal packet filtration.

8.4.2 RQ2: How effective is the proposed method compared to other DoH detection approaches on a real-world ISP dataset?

This research question aimed to validate the accuracy of the approaches when working with real-world data. As shown in Figure 8.3, a real-world dataset for both training and testing (holdout dataset) was used.

Results

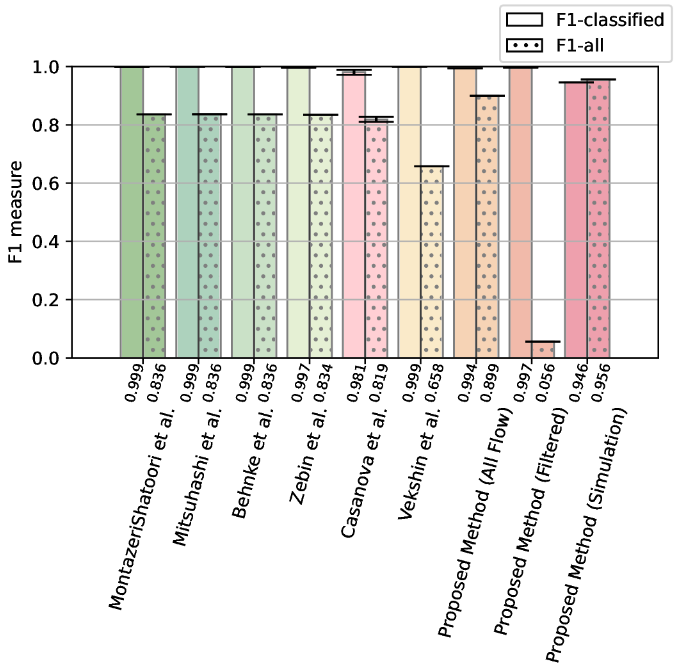


Figure 8.5: RQ2: The performance of DoH detection proposals trained and tested on real-world datasets. The exact measured values of both F1 scores are written under each column. The whiskers in the plot represent the standard deviation observed during different random seeds.

The results of the experiments depicted in Figure 8.5 confirm that the DoH detection proposals achieve excellent performance even on real-world data. Since real-world data contain less variability in the traffic because most of the DoH is serviced by two major DoH resolvers (see Section 6.2), an increase in mean F1-classified scores across most of the proposals can be observed. Compared to generated data, Casanova et al. [21] achieved on real-world data similar accuracy as reported in their study. Only the *proposed method* in the simulation scenario with the active verification performed worse than in the case with generated data.

Such performance drop is caused by the lack of long connections to some DoH resolvers. Due to the condition of minimal packets in the flow (at least 120) for maintaining reasonable accuracy, some flows are directly discarded and classified as non-DoH. Naturally, when there is no flow with at least 120 packets for some resolvers, these resolvers will always be falsely classified as non-DoH.

When looking at the performance measured with F1-all, it can be seen that by far the largest performance drop experienced by the *proposed method* in the Filtered scenario. This is caused by the shape of real-world traffic, where the majority of DoH connections are short. Similarly, Vekshin et al. [129] missed a lot of DoH connections due to his prefiltration to flow with at least five payload packets.

Answer RQ2: This experiment revealed that the DoH detection approaches are again very accurate in real-world datasets. Most of the approaches achieve almost absolute accuracy on F1-classified. The slightly lower mean accuracy is achieved by Casanova et al. [21].

The *proposed approach* Simulated scenario suffers from many short connections that are present in the real network hence lowering accuracy since the short flows are skipped and not passed into the detection module, which is the cause of the lower accuracy of the detection technique. On the other hand, when considering the all-flow scenario of the *proposed method* the accuracy is comparable to the rest of the methods.

Similarly, as in the previous RQ1 when considering F1-all, the methods using the DoHLyzer data source outperform the Vekshin et al. [129] and *proposed method* Filtered since they are filtering more flows before passing into the classifier. However, the simplified features necessary for the *proposed method* all-flow scenario do not need to discard that much of flows in the real-world dataset and hence not lowering the overall accuracy when considering the F1-all case.

Overall, the *proposed method* seems to outperform the other methods, given the simplicity of features needed for an equally accurate method.

8.4.3 RQ3: How effective are DoH detectors trained on the lab-created datasets in real-world ISP deployment?

This research question aims to evaluate the real-world usability of the detectors trained on the laboratory-generated dataset. To simulate this deployment setup, the trained part of the generated dataset was used for training, and all detectors were evaluated using the holdout part from the real-world dataset.

Results

The results are shown in the Figure 8.6. As can be seen, the models trained using the generated dataset are mostly struggling to classify DoH accurately, and most of them are unusable. The only approach that remains usable is the *proposed method* in filtering and simulation scenarios, as it achieves almost the same F1-classified performance as in the previous case. The reason behind such a difference can be attributed to the strict filtering condition of at least 120 packets, where the DoH traffic shape became very distinctive. The traffic shape of shorter DoH flows is highly influenced by the TCP/TLS handshakes or HTTP/2 preface, and it is difficult to recognize DoH from other short HTTPS communication as discussed in previous Chapter 5 and Chapter 7. Since the datasets come from a different environment, the classifiers fail to generalize on short flows and misclassify them.

The approach that also achieved far better performance than the others is the Casanova et al. [21] despite its lowered performance on generated holdout. The neural networks have low explainability, the reasons why it performed this way are unknown and hardly obtainable.

Answer RQ3: As shown in our results, most of the plain ML-based methods are ineffective when trained in one environment and deployed in another. Therefore, the lab-created dataset must be used only as a benchmark and not in production. The deployment into the real world is not concerned by the DoH detector proposals, except for the *proposed method*, which also performed the best in this experiment with the Simulation scenario.

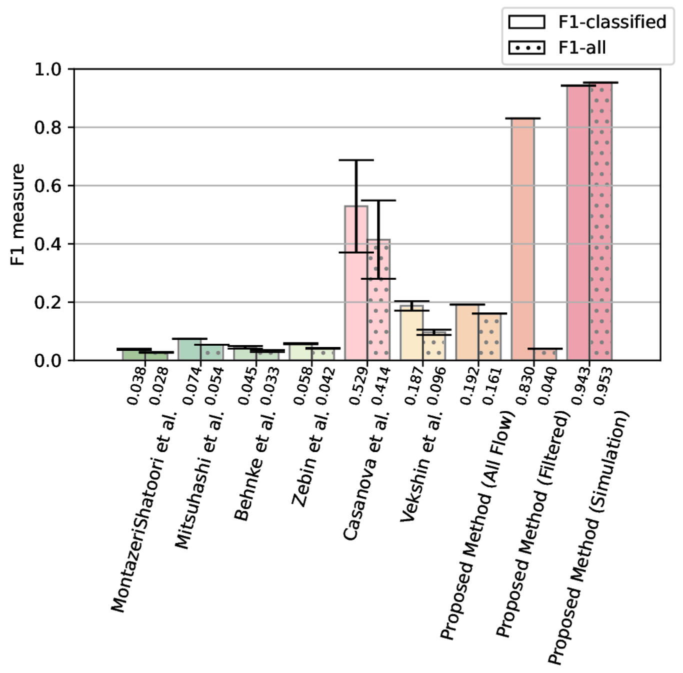


Figure 8.6: RQ3: The performance of DoH detection proposals trained using Generated dataset tested on the real-world dataset. The exact measured values of both F1 scores are written under each column. The whiskers in the plot represent the standard deviation observed during different random seeds.

8.4.4 RQ4: How effective are DoH detectors trained on both types of datasets (lab-created and real-world mixed together) in detecting DoH traffic in real-world ISP deployment?

The research question RQ4 aims to find out the influence of the generated dataset on real-world performance. The real-world dataset contains mainly two DoH resolvers (see Section 6.2). The data are similar to the real-world environment but lack the variability of the generated dataset. Therefore, the rare DoH resolvers might get misclassified as non-DoH.

In this experiment, the training parts consist of a mix of the real world and generated dataset. The whole generated training dataset and 1/2 of the real-world training dataset were used.

From the deployment perspective, this scenario can be seen from two angles. The first covers the case where a laboratory-created dataset exits, and the developers are planning to deploy the detector in the real network hence adding some traffic from the real network to the training set. On the other side, it could represent a case where real network traffic is captured with a limited amount of data, and the developers want to add the laboratory data (data from another environment) to provide more source data, possibly to make the model more robust.

Results

The results of the performed experiments are shown in the in Figure 8.7. As can be seen, the performance of the approaches experienced a small decrease in performance compared to the training on the real-world data only shown in Figure 8.5. The biggest decrease was experienced by the *proposed method* in the all-flow scenario. The drop can be attributed to only four features, which

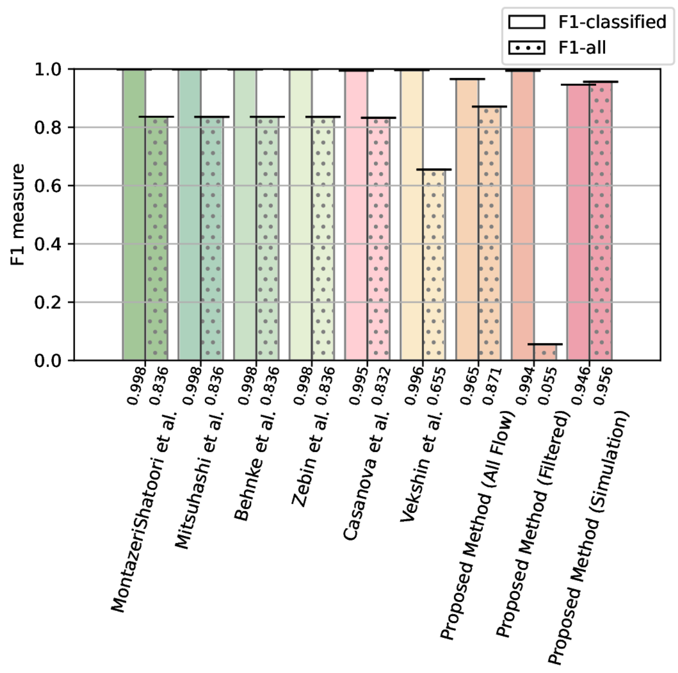


Figure 8.7: RQ4: The performance of DoH detection proposals trained using generated dataset and 1/2 of real-world train dataset. Proposals were tested on the real-world holdout dataset. The exact measured values of both F1 scores are written under each column. The whiskers in the plot represent the standard deviation observed during different random seeds.

limits the capability of covering the nuances of the more complex dataset. However, the simulation scenario proves to be stable compared to training on real-world data only and achieves the same performance. Moreover, Casanova et al. [21] Bi-directional LSTM approach showed increased and more stable performance, which is now comparable to the other methods with the DoHLyzer data source.

Answer RQ4: The experiment showed that the DoH approaches are still very effective. However, the mixture of the Real-World with the Generated dataset does not bring many benefits but rather slightly decreases the performance of most detectors in comparison to the case when trained on the real-world dataset only.

The *proposed method* all-flow scenario decreased performance in this case, but the filtered and simulated scenarios seem to be unchanged.

8.4.5 RQ5: Is data drift a significant phenomenon in the DoH detection task?

Data drift is an important phenomenon that needs to be considered when designing the network detector. The longevity of detectors is an important parameter that depends on various factors, such as the detector design, underlying network infrastructure, or specific time in a year that causes different network traffic shapes or simply changes in the classified service itself.

This research question tests the susceptibility of DoH detectors to data drift on the CESNET2 network backbone environment. During the time between the captures, there was no change in the CESNET2 network or monitoring infrastructure, and thus the data drift originates from the traffic itself. The extreme case is studied here, where the training dataset was captured at least one year before the evaluation data to test the longevity of the detectors truly. Therefore the train part of the real-world dataset was used for training, and for evaluation, the Additional 5-week dataset was used.

Results

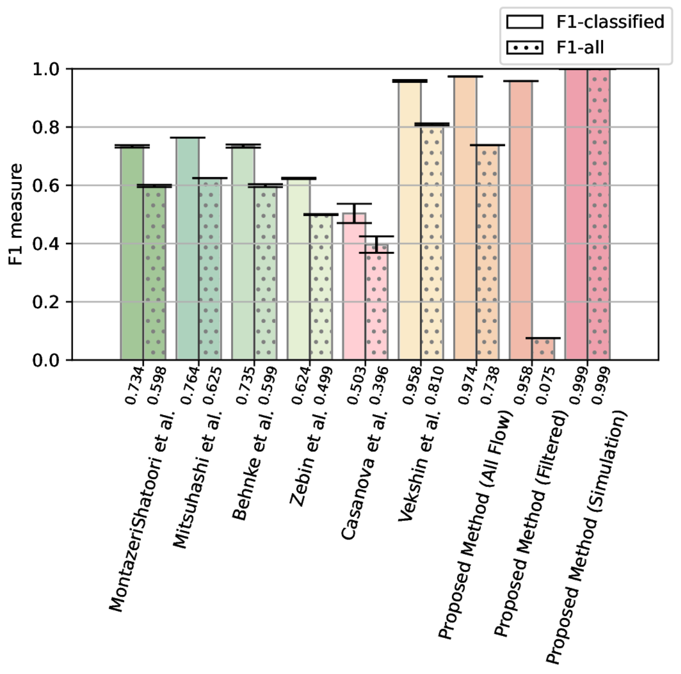


Figure 8.8: RQ5: The performance of proposed detectors trained on the real-world dataset and evaluated on the whole Additional 5-week dataset. The time difference between the training and testing dataset is at least one year.

The results depicted in Figure 8.8 show an expected significant performance drop across all the approaches except the *proposed method* and Vekshin et al. [129] that proved to be more resistant to the longitudinal data drift and suffered only small or no performance degradation. Moreover, the *proposed method* in the simulation case very effectively mitigated the data drift with the adaptive blocklist approach.

Compared to the *proposed method* and Vekshin et al. [129], the approaches using DoHLyzer as a data source proved to be more affected by the longitudinal data drift in both F1-classified and F1-all measures.

Answer RQ5: Despite the extreme case of training and evaluation data captured more than a year apart. It can be concluded that some of the approaches still maintain high accuracy—Vekshin et al. [129] and the *proposed method* achieved an F1-classified score over 0.95. Moreover, the *detection method* in the simulation scenario achieved almost absolute accuracy. Other approaches did not perform that well and are more prone to long-term data drift; however, they still achieved the F1-classified score of over 0.5.

8.4.6 RQ6: How does DoH detection performance degrade over a month in real-world ISP networks?

The previous research question showed the extreme case of long data drift. This research question aims to examine the data drift over a single month. For this experiment, only the Additional 5-week dataset was used. As shown in the Figure 8.3, the training and hyperparameter tuning were performed on the first week, the same way as described in Section 8.3, while the detectors were evaluated on the remaining weeks separately to observe the potential degradation in the classification performance. In the case of the simulation scenario of the *proposed method*, the blocklist was reset so that the evaluation for each weak starts with the empty blocklist. In this experiment, only the F1-classified measure is presented.

Results

The per-week accuracy of each approach is shown in Figure 8.9. The common trend in performance changes. The high performance in Week 2, which then decreases in Week 3. In Week 4, there is a small increase, followed by a more noticeable decrease in Week 5. The last large decrease is common to all measured approaches and can be attributed to Christmas. Since the CESNET2 network is mainly used by universities and research institutions in the Czech Republic, a public holiday such as Christmas often results in significant traffic shape and distribution changes due to a significantly lower amount of transferred data.

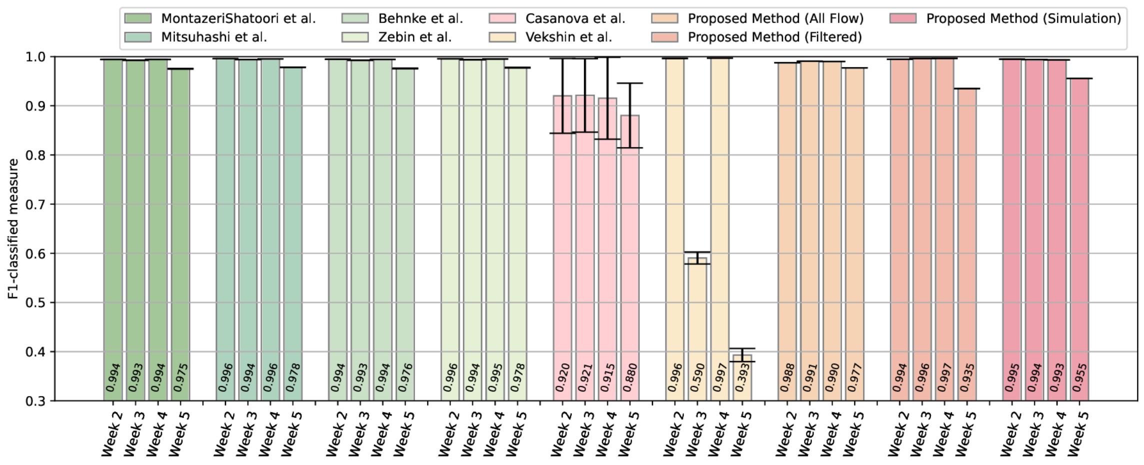


Figure 8.9: RQ6: Per-week F1-classified performance of the DoH detectors trained on the Week 1 of the Additional 5-week dataset. The whiskers in the plot represent the standard deviation observed during different random seeds.

Most of the approaches performed with similar accuracy as reported in the original studies with F1-classified scores over 0.99, with the exception of Vekshin et al. [129] and Casanova et al. [21].

The performance of Vekshin et al. [129] significantly drops in Week 3 and Week 5. This performance instability can be attributed to the smaller training set compared to previous experiments. The smaller set was probably fitted too well, and the model then struggled to deal with the network changes.

Casanova et al. [21], as in previous cases, poses a large variance in the performance depending on the run. The F1-classified score is similar to the one achieved on the generated dataset, which is 0.05 lower than reported in the original study.

Answer RQ6: The experiments show that data drift in the short term does not significantly affect the performance of the classifiers, except for Vekshin et al. [129] and Casanova et al. [21].

The *proposed method* results show similar performance when compared to other methods based on the DoHLyzer data source.

8.5 Discussion

The existing ML-based detectors' performance was evaluated in different scenarios along with the *proposed method*. The best performing across the measured DoH detection proposals are tree-based algorithms, with the exception of Casanova et al. [21], who considered neural networks only. However, the neural network showed a large variability in performance. Figure 8.10 depicts the overall F1-classified performance across all the evaluated cases. The *proposed method* shows the most stable performance with the smallest deviation across all tested scenarios. Generally, the *proposed method* and Vekshin et al. [129] show a lower variance across the experiments. This can be attributed to additional flow filtration that the rest of the methods did not use. The other approaches based on the feature provided by DoHLyzer show similar performance and variance. Nevertheless, the detector proposed by Mitsuhasi et al. [94] achieved a slightly better performance than others.

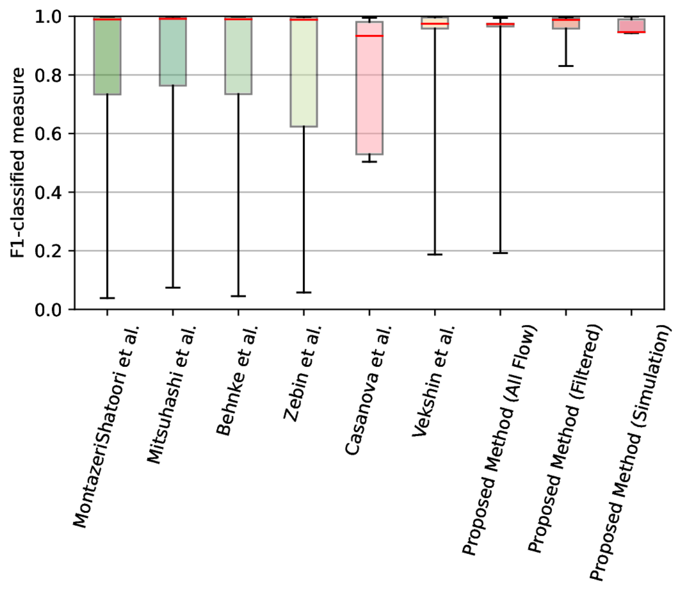


Figure 8.10: Boxplot of the measured mean F1-classified performances for each DoH detection proposal across experiments performed in RQ1–RQ5. The whiskers represent the minimum or maximum observed value.

The results showed that the lab-created datasets can be far more challenging than the real-world ones. The DoH detectors achieved better performance when trained and tested on the real-world dataset compared to the generated one. Nevertheless, most of the ML models proved to be unusable in real-world deployment when trained on the lab-created dataset or generally when trained on the data from one particular computer network and deployed to a different one.

The performance of the models in the RQ3 experiment (trained on the generated and evaluated on real-world datasets) differed vastly. Detectors that performed better in RQ1 and RQ2 (trained and evaluated on similar network data) failed when used with data from different network environments.

More simple but generally less accurate models (in order of 0.01 of F1 score) showed higher transferability. Moreover, they can handle really long time between training and deployment due to lower overfitting and higher generalization.

The data drift (caused by the old training set) can be considered a relatively small problem in the case of DoH detection. Detectors, even one year after training, showed relatively good performance. Nevertheless, most models should be retrained more often—at least once a month to maintain the accuracy over time. Nevertheless, the models should be trained on a sufficient amount of data to mitigate the effect observed with Vekshin et al. [129], which failed to generalize and showed a significant performance variability when trained on data from a single week.

Moreover, the results show that the proposed approaches utilizing a similar set of features extracted by the same DoHLyzer tool have almost the same performance in every test except for Casanova et al. [21]. The approaches focused mostly on algorithm tuning methods with little deviance in the preprocessing phase, such as scaling or sampling involved, bringing almost no benefits. Those can be considered more model-centric approaches. On the other hand, the other approaches, including Vekshin et al. [129] and the *proposed method*, show different characteristics and better generalization, and those can be taken as more data-centric.

The combination of three detection stages of the *proposed method* where the ML-based detector is deployed together with the blocklist shows superior performance over ML-based only proposals. It can detect even single-query flows, which are considered a major challenge by the survey from Hynek et al. [62]. However, small and private DoH resolvers used stealthily (with very short DoH connections, e.g., for Command & Control server access) would still be missed even with current solutions.

Machine learning methods are powerful and can help solve the problems such as DoH detection. However, their sole deployment in computer networks has a downside of high inaccuracy when considering high bandwidth environments, potentially creating an immense amount of false positives and also skipping at least $\sim 25\%$ of the flows that are not covered by the methods. The combination of machine learning methods with other mechanisms using the advantage of the deployment environment can help create a more practical solution.

The *proposed method* relies on a minimal feature vector that is lightweight to compute and can be obtained from any flow monitoring device (including switches and firewalls), which makes the solution far more deployable than the other compared methods. The other methods rely on features that cannot be extracted efficiently in high-speed networks, potentially limiting the applicability of the methods to smaller networks. Moreover, the *proposed method* provides the advantage of maintaining stable high accuracy in environments where frequent retraining is not feasible (e.g., due to privacy policies preventing frequent and automatic traffic captures).

Chapter 9

Conclusion

DNS over HTTPS is a privacy technology that brings encryption to DNS protocol by encapsulating unencrypted DNS messages into encrypted HTTPS. The DoH standardized in October 2018 has already gained adoption among users and service providers. The implementation is present in the majority of web browsers, proxies, and operating systems, and when present, it is enabled as a default option for DNS resolution. It is not the only proposed protocol that aims to bring encryption to DNS; several other protocols exist, such as DNS over TLS and recently standardized DNS over QUIC. Compared to other DNS over Encryption alternatives, DoH is designed to blend into other HTTPS traffic. On one side desired privacy-preserving feature for users, on the other side, it leaves network operators unaware of its presence, representing a potential security risk.

Since its proposal, DoH has been studied by several research teams from various perspectives, including performance measurements, comparing the protocol with other alternatives, and identifying the bottlenecks decreasing resolution efficiency. Privacy and security were other areas that were studied, including correlation attacks, DoH detection, and malicious DoH tunnel detection. Many works were published trying to solve the last two tasks using machine learning methods. However, most of the works only briefly studied the DoH protocol characteristics and rather inclined to a model-centric approach tuning the machine learning detection algorithms on the limited published datasets. The published methods claimed high accuracy but paid little attention to the practicality of the solution utilizing complex attributes, making the solution broadly impractical or very limited.

This work aimed to propose a reliable DoH detection method with a special emphasis on the practicality of the final solution and its broad applicability. The design of the detection method also relies on machine learning, but the work inclines more to a data-centric concept of machine learning. The work focuses on a thorough understanding of the protocol characteristics to create quality and comprehensive datasets and design effective detection method.

At first, the well-known DoH resolvers were studied as a representative sample of one side of the DoH ecosystem. The results showed the security and privacy gaps present in a portion of resolvers that further motivate the regulation of the resolvers by network operators. Moreover, it uncovered characteristics such as header sizes and EDNS padding strategies used by resolvers that create variability in packet sizes influencing the traffic shape of the DoH communication. Gain knowledge was used to improve methodologies and better identify subjects of traffic analysis.

Several measurements filling the gaps of existing works were conducted studying single DoH flows that can be used by small applications such as malware trying to hide their activities or analysis of the traffic characteristics towards multiple resolvers. The single flow analysis uncovered a minimum number of packets necessary to transmit one request/response pair. Analysis of DoH traffic characteristics showed that DoH creates longer connections consisting of more packets with less average size than other HTTPS traffic. DoH resolvers can be distinguished by the sizes given, mainly by the header sizes they are using. The analysis of the most used dataset *CIRA-CIC-DoHBrw-2020* or the other dataset published by Vekshin et al. [128] showed their limited variability and amount of DoH traces that could be improved. The analysis helped determine the characteristics of DoH, and this knowledge was used to create more comprehensive datasets to propose a DoH detection method and to verify it. Several datasets were created, published, and used in this work.

Analysis, datasets, and supportive tools were complementary goals of this thesis, providing a necessary background supporting the main goals, which consist of proposing a reliable detection method and its testing. The proposed method is designed to work with a limited data source extracted by a broadly deployed flow monitoring infrastructure. The machine learning model is combined with filtration and block lists to improve the system's general applicability and higher accuracy. The combination of methods showed to be necessary considering high throughput networks generating high false positive numbers when relying solely on machine learning.

The proposed solution was subject to testing along with other similar state-of-the-art methods in six different scenarios. The methodologies and proposed methods were recreated, trained, and tested on the same comprehensive collection of datasets created in this work to provide a fair comparison of the approaches. The scenarios were designed to test various aspects of the solutions, including real-world applicability, generalization, or longevity of the trained models. The proposed detection method showed the highest stability across all tested scenarios while providing very high accuracy capable of covering all flows. Compared to other approaches, the proposed method uses only four features that are lightweight and can be extracted from most currently deployed monitoring infrastructures, even on high speed and backbone networks. The other approaches are designed to work with complex and hard-to-compute statistical flow features that can not be computed on running sequences, limiting their applicability. Moreover, the other approaches tend to fit the environment well, showing a low generalization capability.

The sole usage of machine learning shows its limits in this network detection task, where even high-accuracy models produce a high amount of false positives over time, potentially overwhelming network operators. The machine learning method combined together with other more exact commonly used approaches in computer networks, including filtration and blocklists, show superior performance when combined together over sole machine learning solution. Moreover, the work shows that domain knowledge helps to create more comprehensive datasets and helps design effective solution. The proposed method solves the problem of DoH detection addressing practicability, lightweigthness, compatibility with existing infrastructures, stability, generalization among networks, and high accuracy, limiting the number of false positives. The method even solves the problem with the detection of short flows. Hence, the solution is considered satisfactory, fulfilling the goals set at the beginning of this thesis.

Bibliography

[1] Aas, J., Barnes, R., Case, B., Durumeric, Z., Eckersley, P. et al. Let's Encrypt: an automated certificate authority to encrypt the entire web. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, p. 2473–2487.

[2] AdGuard. *Known DNS Providers*. Accessed: January 2023. Available at: https://adguard-dns.io/kb/general/dns-providers/.

[3] Affinito, A., Botta, A. and Ventre, G. Local and Public DNS Resolvers: do you trade off performance against security? In: IEEE. *2022 IFIP Networking Conference (IFIP Networking)*. 2022, p. 1–9.

[4] Aitken, P., Claise, B. and Trammell, B. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* [RFC 7011]. 7011. September 2013. Available at: https://www.rfc-editor.org/info/rfc7011.

[5] Alahmadi, B. A., Axon, L. and Martinovic, I. 99% False Positives: A Qualitative Study of {SOC} Analysts' Perspectives on Security Alarms. In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, p. 2783–2800.

[6] Alenezi, R. and Ludwig, S. A. Classifying DNS tunneling tools for malicious DoH traffic. In: IEEE. *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2021, p. 1–9.

[7] authors, T. C. *Chromium code search: doh_provider_entry.cc*. Accessed: June 2021. Available at: https://source.chromium.org/chromium/chromium/src/+/master:net/dns/public/doh_provider_entry.cc.

[8] Baheux, K. *A safer and more private browsing experience on Android with Secure DNS*. Accessed: May 2021. Available at: https://blog.chromium.org/2020/09/a-safer-and-more-private-browsing.html.

[9] Baheux, K. *A safer and more private browsing experience with Secure DNS*. Accessed: May 2021. Available at: https://blog.chromium.org/2020/05/a-safer-and-more-private-browsing-DoH.html.

[10] Banadaki, Y. M. and Robert, S. Detecting malicious dns over https traffic in domain name system using machine learning classifiers. *Journal of Computer Sciences and Applications*. 2020, vol. 8, no. 2, p. 46–55.

[11] Basso, S. Measuring DoT/DoH blocking using OONI probe: a preliminary study. In: *NDSS DNS Privacy Workshop*. 2021.

[12] Behnke, M., Briner, N., Cullen, D., Schwerdtfeger, K., Warren, J. et al. Feature engineering and machine learning model comparison for malicious activity detection in the dns-over-https protocol. *IEEE Access*. IEEE. 2021, vol. 9, p. 129902–129916.

[13] Belshe, M., Peon, R. and Thomson, M. *Hypertext Transfer Protocol Version 2 (HTTP/2)* [RFC 7540]. 7540. May 2015. Available at: https://www.rfc-editor.org/info/rfc7540.

[14] Böttger, T., Cuadrado, F., Antichi, G., Fernandes, E. L., Tyson, G. et al. An Empirical Study of the Cost of DNS-over-HTTPS. In: *Proceedings of the Internet Measurement Conference*. 2019, p. 15–21.

[15] Boutaba, R., Salahuddin, M. A., Limam, N., Ayoubi, S., Shahriar, N. et al. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*. Springer. 2018, vol. 9, no. 1, p. 1–99.

[16] Bumanglag, K. and Kettani, H. On the impact of DNS over HTTPS paradigm on cyber systems. In: IEEE. *2020 3rd International Conference on Information and Computer Technologies (ICICT)*. 2020, p. 494–499.

[17] Burnett, S., Chen, L., Creager, D. A., Efimov, M., Grigorik, I. et al. Network error logging: Client-side measurement of end-to-end web service reliability. In: *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 2020, p. 985–998.

[18] Bushart, J. and Rossow, C. Padding ain't enough: Assessing the privacy guarantees of encrypted {DNS}. In: *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*. 2020.

[19] Callejo, P., Bagnulo, M., Ruiz, J. G., Lutu, A., García Martínez, A. et al. Measuring DoH with web ads. *Computer Networks*. Elsevier. 2022, vol. 212, p. 109046.

[20] Callejo, P., Cuevas, R., Vallina Rodriguez, N. and Rumin, A. C. Measuring the global recursive DNS infrastructure: a view from the edge. *IEEE Access*. IEEE. 2019, vol. 7, p. 168020–168028.

[21] Casanova, L. F. G. and Lin, P.-C. Generalized classification of DNS over HTTPS traffic with deep learning. In: IEEE. *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. 2021, p. 1903–1907.

[22] Chakraborty, G. and Krishna, M. Analysis of unstructured data: Applications of text analytics and sentiment mining. In: *SAS global forum*. 2014, p. 1288–2014.

[23] Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*. 2002, vol. 16, p. 321–357.

[24] Chhabra, R., Murley, P., Kumar, D., Bailey, M. and Wang, G. Measuring DNS-over-HTTPS performance around the world. In: *Proceedings of the 21st ACM Internet Measurement Conference*. 2021, p. 351–365.

[25] Cimpanu, C. *Apple adds support for encrypted DNS (DoH and DoT)*. Accessed: May 2021. Available at: https://www.zdnet.com/article/apple-adds-support-for-encrypted-dns-doh-and-dot/.

[26] Cisco. *Umbrella Popularity List*. Accessed: May 2019. Available at: http://s3-us-west-1.amazonaws.com/umbrella-static/top-1m.csv.zip.

[27] Cisco Systems, Inc.. *NetFlow Export Datagram Format*. September 2007. Accessed: December 2022. Available at: https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html.

[28] Claise, B. *Cisco Systems NetFlow Services Export Version 9* [RFC 3954]. 3954. October 2004. Available at: https://doi.org/10.17487/RFC3954.

[29] Cloudflare. *Using JSON*. Accessed: May 2021. Available at: https://developers.cloudflare.com/1.1.1.1/dns-over-https/json-format/.

[30] Crichton, K., Christin, N. and Cranor, L. F. How do home computer users browse the web? *ACM Transactions on the Web (TWEB)*. ACM New York, NY. 2021, vol. 16, no. 1, p. 1–27.

[31] Csikor, L., Singh, H., Kang, M. S. and Divakaran, D. M. Privacy of DNS-over-HTTPS: Requiem for a Dream? In: IEEE. *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2021, p. 252–271.

[32] Dahanayaka, T., Wang, Z., Jourjon, G. and Seneviratne, S. Inline Traffic Analysis Attacks on DNS over HTTPS. In: IEEE. *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. 2022, p. 132–139.

[33] Damas, J., Graff, M. and Vixie, P. *Extension Mechanisms for DNS (EDNS(0))*. Datatracker.Ietf.Org, 2013. Available at: https://datatracker.ietf.org/doc/html/rfc6891.

[34] David P. Wiggins, I. *XVFB*. 2010. Accessed: September 2021. Available at: https://www.x.org/releases/X11R7.6/doc/man/man1/Xvfb.1.xhtml.

[35] Deccio, C. and Davis, J. DNS privacy in practice and preparation. In: *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*. 2019, p. 138–143.

[36] Deckelmann, S. *Firefox continues push to bring DNS over HTTPS by default for US users*. Accessed: May 2021. Available at: https://blog.mozilla.org/blog/2020/02/25/firefox-continues-push-to-bring-dns-over-https-by-def\ault-for-us-users/.

[37] Dierks, T. and Rescorla, E. The transport layer security (TLS) protocol version 1.2. RFC 5246, August. 2008.

[38] Ding, S., Zhang, D., Ge, J., Yuan, X. and Du, X. Encrypt DNS traffic: automated feature learning method for detecting DNS tunnels. In: IEEE. *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. 2021, p. 352–359.

[39] DNSCrypt. *List of public DoH and DNSCrypt servers*. Accessed: January 2023. Available at: https://dnscrypt.info/public-servers/.

[40] Doan, T. V., Fries, J. and Bajpai, V. Evaluating public DNS services in the wake of increasing centralization of DNS. In: IEEE. *2021 IFIP Networking Conference (IFIP Networking)*. 2021, p. 1–9.

[41] Du, X., Liu, D., Ding, S., Liu, Z., Yuan, X. et al. Design of an Autoencoder-based Anomaly Detection for the DoH traffic System. In: IEEE. *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. 2022, p. 763–768.

[42] Federrath, H., Fuchs, K.-P., Herrmann, D. and Piosecny, C. Privacy-preserving DNS: analysis of broadcast, range queries and mix-based protection methods. In: Springer. *Computer Security–ESORICS 2011: 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings 16*. 2011, p. 665–683.

[43] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L. et al. RFC 2616: Hypertext transfer protocol–HTTP/1.1, June 1999. *Status: Standards Track*. 1999, vol. 1, no. 11, p. 1829–1841.

[44] García, S., Bogado, J., Hynek, K., Vekshin, D., Čejka, T. et al. Large scale analysis of doh deployment on the internet. In: Springer. *European Symposium on Research in Computer Security*. 2022, p. 145–165.

[45] García, S., Hynek, K., Vekshin, D., Čejka, T. and Wasicek, A. Large scale measurement on the adoption of encrypted DNS. *ArXiv preprint arXiv:2107.04436*. 2021.

[46] Google. *Privacy best practices*. Accessed: September 2021. Available at: https://developers.google.com/speed/public-dns/docs/doh#privacy_best_practices.

[47] Han, J., Pei, J. and Tong, H. *Data mining: concepts and techniques*. Morgan Kaufmann, 2022.

[48] Harris, G. and Richardson, M. *PCAP Capture File Format*. Internet-Draft draft-gharris-opsawg-pcap-02. Internet Engineering Task Force, june 2021. Work in Progress. Available at: https://datatracker.ietf.org/doc/html/draft-gharris-opsawg-pcap-02.

[49] Hoffman, P. *Representing DNS Messages in JSON*. Datatracker.Ietf.Org, 2020. Available at: https://datatracker.ietf.org/doc/rfc8427/.

[50] Hoffman, P. and McManus, P. *History For Draft IETF DOH-DNS-Over-HTTPS-14*. Datatracker.Ietf.Org, 2019. Available at: https://datatracker.ietf.org/doc/rfc8484/history/.

[51] Hoffman, P. E. and McManus, P. *DNS Queries over HTTPS (DoH)* [RFC 8484]. 8484. October 2018.

[52] Hofstede, R., Bartoș, V., Sperotto, A. and Pras, A. Towards real-time intrusion detection for NetFlow and IPFIX. In: IEEE. *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*. 2013, p. 227–234.

[53] Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R. et al. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys & Tutorials*. IEEE. 2014, vol. 16, no. 4, p. 2037–2064.

[54] Hounsel, A., Borgolte, K., Schmitt, P., Holland, J. and Feamster, N. Analyzing the costs (and benefits) of DNS, DoT, and DoH for the modern web. In: *Proceedings of the applied networking research workshop*. 2019, p. 20–22.

[55] Hounsel, A., Borgolte, K., Schmitt, P., Holland, J. and Feamster, N. Comparing the effects of DNS, DoT, and DoH on web performance. In: *Proceedings of The Web Conference 2020*. 2020, p. 562–572.

[56] Hounsel, A., Schmitt, P., Borgolte, K. and Feamster, N. Can encrypted dns be fast? In: Springer. *Passive and Active Measurement: 22nd International Conference, PAM 2021, Virtual Event, March 29–April 1, 2021, Proceedings 22*. 2021, p. 444–459.

[57] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D. et al. Specification for dns over transport layer security (tls). *IETF RFC7858, May*. 2016.

[58] Huang, Q., Chang, D. and Li, Z. A Comprehensive Study of {DNS-over-HTTPS} Downgrade Attack. In: *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*. 2020.

[59] Huc, M. *How to enable DNS over HTTPS (DoH) on Windows 11*. November 2022. Accessed: December 2022. Available at: https://pureinfotech.com/enable-dns-over-https-windows-11/.

[60] Huitema, C., Dickinson, S. and Mankin, A. *DNS over Dedicated QUIC Connections*. Datatracker.Ietf.Org, 2022. Available at: https://datatracker.ietf.org/doc/rfc9250/.

[61] Hynek, K. and Cejka, T. Privacy illusion: Beware of unpadded DoH. In: IEEE. *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. 2020, p. 0621–0628.

[62] Hynek, K., Vekshin, D., Luxemburk, J., Cejka, T. and Wasicek, A. Summary of DNS over https abuse. *IEEE Access*. IEEE. 2022, vol. 10, p. 54668–54680.

[63] Iyengar, J. and Thomson, M. RFC 9000 QUIC: A UDP-based multiplexed and secure transport. *Omtermet Emgomeeromg Task Force*. 2021.

[64] Jarrahi, M. H., Memariani, A. and Guha, S. The Principles of Data-Centric AI. *Communications of the ACM*. ACM New York, NY, USA. 2023, vol. 66, no. 8, p. 84–92.

[65] Jeřábek, K., Hynek, K., Čejka, T. and Ryşavỳ, O. Collection of datasets with DNS over HTTPS traffic. *Data in Brief*. Elsevier. 2022, vol. 42, p. 108310.

[66] Jerabek, K., Hynek, K., Rysavy, O. and Burgetova, I. DNS over HTTPS Detection Using Standard Flow Telemetry. *IEEE Access*. IEEE. 2023.

[67] Jeřábek, K. and Ryşavỳ, O. Big data network flow processing using Apache Spark. In: *Proceedings of the 6th conference on the engineering of computer based systems*. 2019, p. 1–9.

[68] Jerabek, K., Rysavy, O. and Burgetova, I. Measurement and characterization of DNS over HTTPS traffic. *ArXiv preprint arXiv:2204.03975*. 2022.

[69] Jerabek, K., Rysavy, O. and Burgetova, I. Analysis of Well-Known DNS over HTTPS Resolvers. In: IEEE. *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*. 2023, p. 0516–0524.

[70] Jeřábek, K., Hynek, K. and Ryşavý, O. *Five-week DoH Dataset collected on ISP backbone lines*. Zenodo, september 2023. DOI: 10.5281/zenodo.8348773. Available at: https://doi.org/10.5281/zenodo.8348773.

[71] Jeřábek, K. and Stuchlý, S. *DNS Over HTTPS network traffic*. IEEE Dataport, 2021. DOI: 10.21227/96ea-2055. Available at: https://dx.doi.org/10.21227/96ea-2055.

[72] Jha, H., Patel, I., Li, G., Cherukuri, A. K. and Thaseen, S. Detection of Tunneling in DNS over HTTPS. In: IEEE. *2021 7th International Conference on Signal Processing and Communication (ICSC)*. 2021, p. 42–47.

[73] Khodjaeva, Y., Zincir Heywood, N. and Zincir, I. Can We Detect Malicious Behaviours in Encrypted DNS Tunnels Using Network Flow Entropy? *Journal of Cyber Security and Mobility*. 2022, p. 461–496.

[74] Kinnear, E., McManus, P., Pauly, T., Verma, T. and Wood, C. *Oblivious DNS over HTTPS*. Datatracker.Ietf.Org, 2022. Available at: https://datatracker.ietf.org/doc/rfc9230/.

[75] Konopa, M., Fesl, J., Jelínek, J., Feslová, M., Cehák, J. et al. Using machine learning for DNS over HTTPS detection. In: *Proc. 19th Eur. Conf. Cyber Warfare*. 2020, p. 205.

[76] Kosek, M., Doan, T. V., Granderath, M. and Bajpai, V. One to rule them all? a first look at dns over quic. In: Springer. *International Conference on Passive and Active Network Measurement*. 2022, p. 537–551.

[77] Kosek, M., Schumann, L., Marx, R., Doan, T. V. and Bajpai, V. DNS privacy with speed? evaluating DNS over QUIC and its impact on web performance. In: *Proceedings of the 22nd ACM Internet Measurement Conference*. 2022, p. 44–50.

[78] Kwan, C., Janiszewski, P., Qiu, S., Wang, C. and Bocovich, C. Exploring simple detection techniques for DNS-over-HTTPS tunnels. In: *Proceedings of the ACM SIGCOMM 2021 Workshop on Free and Open Communications on the Internet*. 2021, p. 37–42.

[79] LeBlanc, B. *Announcing Windows 10 Insider Preview Build 19628*. Accessed: May 2021. Available at: https://blogs.windows.com/windows-insider/2020/05/13/announcing-windows-10-insider-preview-build-19628/.

[80] Li, B., He, S., Peng, H., Zhang, E. and Xin, J. Detecting DoH tunnels with privacy protection using federated learning. In: SPIE. *International Conference on Network Communication and Information Security (ICNIS 2021)*. 2022, vol. 12175, p. 133–141.

[81] Li, Y., Dandoush, A. and Liu, J. Evaluation and Optimization of learning-based DNS over HTTPS Traffic Classification. In: IEEE. *2021 International Symposium on Networks, Computers and Communications (ISNCC)*. 2021, p. 1–6.

[82] Lipton, Z. C. and Steinhardt, J. Research for practice: troubling trends in machine-learning scholarship. *Communications of the ACM*. ACM New York, NY, USA. 2019, vol. 62, no. 6, p. 45–53.

[83] Lu, C., Liu, B., Li, Z., Hao, S., Duan, H. et al. An end-to-end, large-scale measurement of dns-over-encryption: How far have we come? In: *Proceedings of the Internet Measurement Conference*. 2019, p. 22–35.

[84] Luo, M., Yao, Y., Xin, L., Jiang, Z., Wang, Q. et al. Measurement for encrypted open resolvers: Applications and security. *Computer Networks*. Elsevier. 2022, vol. 213, p. 109081.

[85] Luxemburk, J. and Čejka, T. Fine-grained TLS services classification with reject option. *Computer Networks*. Elsevier. 2023, vol. 220, p. 109467.

[86] Lyu, M., Gharakheili, H. H. and Sivaraman, V. A survey on DNS encryption: Current development, malware misuse, and inference techniques. *ACM Computing Surveys*. ACM New York, NY. 2022, vol. 55, no. 8, p. 1–28.

[87] Majestic. *The Majestic Million*. 2021. Accessed: July 2021. Available at: https://majestic.com/reports/majestic-million.

[88] Malekghaini, N., Akbari, E., Salahuddin, M. A., Limam, N., Boutaba, R. et al. Deep learning for encrypted traffic classification in the face of data drift: An empirical study. *Computer Networks*. Elsevier. 2023, vol. 225, p. 109648.

[89] Mayrhofer, A. *The EDNS(0) Padding Option*. Datatracker.Ietf.Org, 2016. Available at: https://datatracker.ietf.org/doc/html/rfc7830.

[90] Mayrhofer, A. Padding Policies for Extension Mechanisms for DNS (EDNS (0)). *Internet Requests for Comments, IETF, RFC*. 2018, vol. 8467.

[91] Mbewe, E. S. and Chavula, J. On QoE impact of DoH and DoT in Africa: Why a user's DNS choice matters. In: Springer. *Towards new e-Infrastructure and e-Services for Developing Countries: 12th EAI International Conference, AFRICOMM 2020, Ebène City, Mauritius, December 2-4, 2020, Proceedings 12*. 2021, p. 289–304.

[92] Mitchell, T. M. *Machine learning*. 1997.

[93] Mitsuhashi, R., Jin, Y., Iida, K., Shinagawa, T. and Takai, Y. Malicious DNS Tunnel Tool Recognition using Persistent DoH Traffic Analysis. *IEEE Transactions on Network and Service Management*. IEEE. 2022.

[94] Mitsuhashi, R., Jin, Y., Iida, K., Shinagawa, T. and Takai, Y. Detection of DGA-based Malware Communications from DoH Traffic Using Machine Learning Analysis. In: IEEE. *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*. 2023, p. 224–229.

[95] Mockapetris, P. *Domain names - implementation and specification* [RFC 1035]. 1035. November 1987. Available at: https://www.rfc-editor.org/info/rfc1035.

[96] MontazeriShatoori, M., Davidson, L., Kaur, G. and Lashkari, A. H. *CIRA-CIC-DoHBrw-2020*. Accessed: May 2022. Available at: https://www.unb.ca/cic/datasets/dohbrw-2020.html.

[97] MontazeriShatoori, M., Davidson, L., Kaur, G. and Lashkari, A. H. Detection of doh tunnels using time-series classification of encrypted traffic. In: IEEE. *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. 2020, p. 63–70.

[98] Moore, A., Zuev, D. and Crogan, M. *Discriminators for use in flow-based classification*. 2013.

[99] Moore, A. W. and Zuev, D. Internet traffic classification using bayesian analysis techniques. In: *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. 2005, p. 50–60.

[100] Moure Garrido, M., Campo, C. and Garcia Rubio, C. Detecting Malicious Use of DoH Tunnels Using Statistical Traffic Analysis. In: *Proceedings of the 19th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*. 2022, p. 25–32.

[101] Mozilla. *Bug 1543811: EDNS Padding support for encrypted DNS transports*. Accessed: May 2021. Available at: https://bugzilla.mozilla.org/show_bug.cgi?id=1543811.

[102] Mühlhauser, M., Pridöhl, H. and Herrmann, D. How private is Android's private DNS setting? Identifying apps by encrypted DNS traffic. In: *Proceedings of the 16th International Conference on Availability, Reliability and Security*. 2021, p. 1–10.

[103] Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[104] Namdev, N., Agrawal, S. and Silkari, S. Recent advancement in machine learning based internet traffic classification. *Procedia Computer Science*. Elsevier. 2015, vol. 60, p. 784–791.

[105] Nguyen, T. A. and Park, M. Doh tunneling detection system for enterprise network using deep learning technique. *Applied Sciences*. MDPI. 2022, vol. 12, no. 5, p. 2416.

[106] Niakanlahiji, A., Orlowski, S., Vahid, A. and Jafarian, J. H. Toward practical defense against traffic analysis attacks on encrypted DNS traffic. *Computers & Security*. Elsevier. 2023, vol. 124, p. 103001.

[107] Nijeboer, F. *Detection of https encrypted dns traffic*. 2020. B.S. thesis. University of Twente.

[108] Nisenoff, A., Feamster, N., Hoofnagle, M. A. and Zink, S. User expectations and understanding of encrypted DNS settings. In: *Proc. NDSS DNS Privacy Workshop. Virtual Event*. 2021.

[109] OWASP. *OWASP Cheat Sheet Series*. Accessed: January 2023. Available at: https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html.

[110] Peon, R. and Ruellan, H. *HPACK: Header compression for HTTP/2*. 2015.

[111] Praseed, A. and Thilagam, P. S. Multiplexed asymmetric attacks: Next-generation DDoS on HTTP/2 servers. *IEEE Transactions on Information Forensics and Security*. IEEE. 2019, vol. 15, p. 1790–1800.

[112] Qiu, Y., Li, B., Jiao, L., Zhu, Y. and Liu, Q. Detection of DoH Tunnels with Dual-Tier Classifier. In: IEEE. *2022 18th International Conference on Mobility, Sensing and Networking (MSN)*. 2022, p. 417–421.

[113] Quad9 Foundation. *IOS and MacOS Mobile Provisioning Profiles Are Here!* March 2022. Accessed: December 2022. Available at: https://www.quad9.net/news/blog/ios-mobile-provisioning-profiles/.

[114] Rescorla, E. et al. Rfc 8446: The transport layer security (tls) protocol version 1.3. *Internet Engineering Task Force (IETF)*. 2018, p. 25.

[115] Roach, A. *Trusted Recursive Resolver*. Accessed: May 2021. Available at: https://wiki.mozilla.org/Trusted_Recursive_Resolver.

[116] Sabilla, S. I., Sarno, R. and Triyana, K. Optimizing threshold using pearson correlation for selecting features of electronic nose signals. *Int. J. Intell. Eng. Syst.* 2019, vol. 12, no. 6, p. 81–90.

[117] Salsabila, H., Mardhiyah, S. and Hadiprakoso, R. B. Flubot Malware Hybrid Analysis on Android Operating System. In: IEEE. *2022 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*. 2022, p. 202–206.

[118] Sculley, D., Snoek, J., Wiltschko, A. and Rahimi, A. Winner's curse? On pace, progress, and empirical rigor. 2018.

[119] Siby, S., Juarez, M., Diaz, C., Vallina Rodriguez, N. and Troncoso, C. Encrypted DNS–> Privacy? A traffic analysis perspective. *ArXiv preprint arXiv:1906.09682*. 2019.

[120] Singanamalla, S., Chunhapanya, S., Vavruşa, M., Verma, T., Wu, P. et al. Oblivious dns over https (odoh): A practical privacy enhancement to dns. *ArXiv preprint arXiv:2011.10121*. 2020.

[121] Singh, P. Systematic review of data-centric approaches in artificial intelligence and machine learning. *Data Science and Management*. Elsevier. 2023.

[122] Smith, P. G. *Professional website performance: optimizing the front-end and back-end*. John Wiley & Sons, 2012.

[123] Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A. et al. An overview of IP flow-based intrusion detection. *IEEE communications surveys & tutorials*. IEEE. 2010, vol. 12, no. 3, p. 343–356.

[124] Steadman, J. and Scott Hayward, S. Detecting data exfiltration over encrypted dns. In: IEEE. *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. 2022, p. 429–437.

[125] Steinberger, J., Schehlmann, L., Abt, S. and Baier, H. Anomaly Detection and mitigation at Internet scale: A survey. In: Springer. *Emerging Management Mechanisms for the Future Internet: 7th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2013, Barcelona, Spain, June 25-28, 2013. Proceedings 7*. 2013, p. 49–60.

[126] Stenberg, D. *DNS over HTTPS · curl/curl Wiki*. Accessed: May 2021. Available at: https://github.com/curl/curl/wiki/DNS-over-HTTPS.

[127] Trevisan, M., Soro, F., Mellia, M., Drago, I. and Morla, R. Attacking DoH and ECH: Does server name encryption protect users' privacy? *ACM Transactions on Internet Technology*. ACM New York, NY. 2023, vol. 23, no. 1, p. 1–22.

[128] Vekshin, D., Hynek, K. and Cejka, T. *Dataset used for detecting DNS over HTTPS by Machine Learning*. Zenodo, may 2020. DOI: 10.5281/zenodo.3906526. Available at: https://doi.org/10.5281/zenodo.3906526.

[129] Vekshin, D., Hynek, K. and Cejka, T. Doh insight: Detecting dns over https by machine learning. In: *Proceedings of the 15th International Conference on Availability, Reliability and Security*. 2020, p. 1–8.

[130] Veselỳ, V. and Žádník, M. How to detect cryptocurrency miners? By traffic forensics! *Digital Investigation*. Elsevier. 2019, vol. 31, p. 100884.

[131] VMware, Inc. *TCP segmentation offload*. 2020. Available at: https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.networking.doc/GUID-E105A601-9331-496C-A213-F76EA3863E31.html.

[132] Vyas, K. *Android 13 finally adds native support for DNS over HTTPS*. April 2022. Accessed: November 2022. Available at: https://www.xda-developers.com/android-13-dns-https-support/.

[133] W3C. *Network Error Logging*. Accessed: January 2023. Available at: https://w3c.github.io/network-error-logging/#privacy-considerations.

[134] W3C. *Reporting API*. Accessed: January 2023. Available at: https://w3c.github.io/reporting/#privacy.

[135] Wang, Y., Xiang, Y. and Yu, S. Internet traffic classification using machine learning: a token-based approach. In: IEEE. *2011 14th IEEE International Conference on Computational Science and Engineering*. 2011, p. 285–289.

[136] Wu, J., Zhu, Y., Li, B., Liu, Q. and Fang, B. Peek inside the encrypted world: Autoencoder-based detection of doh resolvers. In: IEEE. *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2021, p. 783–790.

[137] Yan, Z. and Lee, J.-H. The road to DNS privacy. *Future Generation Computer Systems*. Elsevier. 2020, vol. 112, p. 604–611.

[138] Zebin, T., Rezvy, S. and Luo, Y. An explainable AI-based intrusion detection system for DNS over HTTPS (DoH) attacks. *IEEE Transactions on Information Forensics and Security*. IEEE. 2022, vol. 17, p. 2339–2349.

[139] Zhan, M., Li, Y., Yu, G., Li, B. and Wang, W. Detecting DNS over HTTPS based data exfiltration. *Computer Networks*. Elsevier. 2022, vol. 209, p. 108919.

[140] Zhou, W., Dong, L., Bic, L., Zhou, M. and Chen, L. Internet traffic classification using feed-forward neural network. In: IEEE. *2011 International conference on computational problem-solving (ICCP)*. 2011, p. 641–646.

[141] Zou, F., Meng, D., Gao, W. and Li, L. DePL: Detecting Privacy Leakage in DNS-over-HTTPS Traffic. In: IEEE. *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2021, p. 577–586.