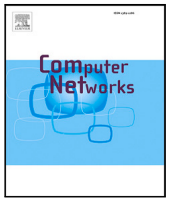
Computer Networks 209 (2022) 108919

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Detecting DNS over HTTPS based data exfiltration

Mengqi Zhan [a,b], Yang Li [a,b,*], Guangxi Yu [a], Bo Li [a,b], Weiping Wang [a,b]

[a] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[b] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

ARTICLE INFO

Keywords:
DNS over HTTPS
DNS tunneling
Data exfiltration
Detection
TLS fingerprinting

ABSTRACT

DNS is often used by attackers as a covert channel for data exfiltration, also known as DNS tunneling. Since the plaintext DNS lookup leads to privacy issues, DNS over HTTPS (DoH) has recently been standardized and deployed. DoH encapsulates DNS in HTTPS to encrypt DNS traffic between clients and recursive resolvers. Attackers can also use DoH for subtle data exfiltration. However, existing DNS tunneling detection methods for plaintext DNS are usually ineffective for DoH tunneling. In this paper, we propose a method to detect DoH-based data exfiltration. We analyze TLS fingerprints of DoH clients and build classifiers with flow-based features to detect DoH tunneling. Our experiment discusses the influence of various factors on the detection results in detail, including adversarial considerations by exploring the potential evasion. Experimental results demonstrate that the proposed method is effective, and it is very difficult to evade detection due to the difficulty of feature imitation. Besides, our method can still provide the defender with helpful information for attack investigation even if the attacker evades detection.

1. Introduction

The Domain Name System (DNS) that maps domain names and IP addresses to each other is the cornerstone of the Internet. Security facilities such as firewalls generally do not block DNS packets due to the necessity of DNS, which means that DNS is often used by attackers as a covert channel, also known as DNS tunneling. DNS tunneling is widely used to transfer stolen information. Both individual users and enterprises may be the targets of data breach attacks. Such attacks may be aimed at compromising the privacy of a user, or even an advanced persistent threat (APT) level attack aimed at stealing more advanced secrets such as trade secrets. High-profile data breaches through DNS are reported almost every year, such as Backdoor.Win32.Denis [1] in 2017 and UDPoS [2] in 2018. Detecting DNS tunneling has attracted much attention from researchers in the past decade. The detection is generally based on the structural, linguistics, and statistics features of the domain name [3–7]. Such methods are usually based on a premise that DNS is not considered to be encrypted as the DNS standard proposed. DNS messages are transmitted in plain text, allowing any third party to monitor DNS traffic and obtain domain information.

Although the plaintext transmission mechanism of DNS facilitates security inspections, it obviously leads to serious privacy issues. Several studies have demonstrated that DNS traffic is used for performing censorship and surveillance [8–10]. This situation has led to the development of secure protocols for providing privacy for DNS. The DNS over HTTPS (DoH) protocol was officially standardized in October 2018 [11]. DoH aims to perform DNS queries and responses through encrypted HTTPS requests, thus preventing plaintext DNS queries and responses from being eavesdropped or tampered with by third parties. At present, DoH has been supported by representative public DNS resolvers such as Cloudflare and Google. Browsers such as Chrome and Firefox have also added DoH support to stable releases as of this writing. It can be predicted that the proportion of DoH will gradually increase in the future.

DoH is a great step for privacy. However, it also means that DNS-based malicious activities can hide through encryption. Due to the loss of visibility to DNS queries and responses (that is, the inability to know the content of specific fields in DNS queries or responses), most existing methods for detecting DNS covert channels based on domain features will be invalid. As a relatively new protocol, the security and privacy-related issues of DoH are still being explored by researchers. At present, relevant studies mainly focus on distinguishing DNS traffic and encrypted DNS traffic [12]; analyzing the effectiveness of encrypted DNS for privacy protection such as web fingerprinting from encrypted DNS traffic [13–16]. Unfortunately, attackers have begun to use DoH as a covert channel for data exfiltration in the wild. According to the report, Godlua is the first-ever malware strain seen using DoH to hide its DNS traffic in 2019 [17], and the first known APT organization using DoH as an exfiltration channel appeared in 2020 [18].

To fill this gap, in this paper, we propose a novel method to detect DoH-based data exfiltration (i.e., DoH tunneling). We focus on

* Corresponding author at: Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China.
E-mail addresses: zhanmengqi@iie.ac.cn (M. Zhan), liyang@iie.ac.cn (Y. Li).

https://doi.org/10.1016/j.comnet.2022.108919
Received 18 August 2021; Received in revised form 28 February 2022; Accepted 14 March 2022
Available online 23 March 2022

the traffic between the victim and the recursive resolver similar to the previous works [5,6,19]. The traffic-based detection method is transparent to the host. The administrator can monitor the traffic at the network boundary to protect all the hosts within the organization and avoid the heavy burden of deploying security protection solutions on all hosts. We hope to build an effective classifier to distinguish DoH tunneling traffic from benign DoH traffic. To do that, we need to extract features from encrypted DoH traffic that can be used to detect DoH tunneling traffic. Since the DoH wraps the DNS traffic in HTTPS, an observed DoH flow can be divided into two phases. The first phase is generated by the HTTPS client–server handshake process. In the second phase, the client and server send encrypted data packets to each other, which contain DNS queries/responses. Our insight is to extract features that can detect DoH tunneling traffic from these two phases. The HTTPS client–server handshake process refers to the process of exchanging keys between the client and server to establish a connection [20]. The first few packets sent by the client and server during the handshake process contain plaintext information, and this information can be used to recognize which application the connection comes from. Recent works have discussed the use of such a technique (called TLS fingerprinting) for malware detection [20–22]. Inspired by these works, we measure TLS fingerprints of clients that currently support DoH, and propose to establish a whitelist mechanism to preliminarily detect DoH tunneling based on TLS fingerprints. For encrypted traffic, statistical features are considered to be very powerful in performing analyses. We analyze the features of DoH tunneling, extract flow-based features from encrypted traffic that contain DNS queries/responses, and build classifiers to detect DoH tunneling.

We design a series of experiments to verify the effectiveness of the proposed method. Based on the data collected in the Internet environment, we find that differences between the training and deployment environment (e.g., victim location, choice of recursive resolver) can affect detection performance. We also discuss in detail how attackers might evade detection. Since domain-related features are very helpful for detecting DNS tunneling, making DNS queries/responses invisible will intuitively degrade detection performance. In addition to directly testing the performance of the classifier on the idealized dataset, we also simulate scenarios that are conducive to the attacker to evade detection in experiments, hoping to find the *lower bound* of the detection ability of the proposed method. Our insight is that if the attacker needs to pay an extremely high cost (even only theoretically feasible) to evade the proposed method, the proposed method is effective for DoH tunneling detection. Our experimental results show that the proposed method can effectively detect DoH tunneling in most cases. Even if the attacker evades detection (which is very difficult), the transmission rate of the DoH tunnel will be very low, and the proposed method can still provide the defender with helpful information for attack investigation.

Our main contributions of this paper are briefly summarized as follows:

- We propose a novel method to detect DoH-based data exfiltration, aiming to address the challenge that most existing methods of detecting DNS tunneling are invalid for DoH tunneling since DNS requests and responses are encrypted by DoH.
- Our proposed method includes two detection steps, which act on the TLS handshake phase and the data transmission phase of the DoH flow respectively. TLS fingerprints of DoH clients are first used for providing the preliminary warning, and then flow-based features of the DoH flow are extracted and used by classifiers to accurately detect DoH tunneling.
- We conduct extensive experiments with the data collected on the Internet. Our experiments simulate various scenarios, such as the location of the victim and the choice of DoH resolver, and discuss the impact of these factors on detection. Furthermore, our experiments include adversarial considerations and discuss ways that attackers might use to evade detection. The experimental results indicate the effectiveness of the proposed method.

The rest of the paper is organized as follows. The background is presented in Section 2. The threat model and overview of the proposed detection method are described in Section 3. The introduction of the TLS fingerprint measurement of DoH clients is presented in Section 4. The design of flow-based detection methods is presented in Section 5. A series of experiments on flow-based detection are presented in Section 6. Discussion and related works are addressed in Section 7 and Section 8, respectively. Finally, the paper is concluded in Section 9.

2. Background

2.1. The Domain Name System

The Domain Name System (DNS) resolves easy-to-read domain names into numerical IP addresses. Typically, to resolve the domain name, the client sends a DNS query to the recursive resolver. If the recursive resolver does not cache the domain name of the query, it will recursively communicate with the authoritative name server from the root name server until the resolver has an authoritative answer to the user's query. The client can use the resolved IP address to connect to the target host. Recursive resolvers are usually provided by the Internet service provider or organizations to serve multiple users.

2.2. Data exfiltration over DNS

If the host is connected to the World Wide Web, then the firewall is usually configured to allow packets on UDP port 53 (used by DNS) to be sent to the internal DNS server, or even to allow all packets on UDP port 53 directly since DNS is such a crucial service for virtually all applications. Therefore, DNS is an ideal covert channel for attackers. DNS covert channel, or DNS tunneling, essentially encodes and encapsulates data in DNS queries. Specifically, the data is encoded into the subdomain of the target domain name. To receive the data, the attacker needs to take over the Name Server (NS) of the target domain name, so that all subdomain resolution requests of the target domain name will eventually reach the controlled NS. The subdomain of the target domain name (i.e., the encoded information) is not duplicated over a period of time (within the time-to-live) so that the recursive resolver cannot find the domain name in the cache. In this way, the domain name query is always forwarded to the controlled NS. The tool on the controlled NS decodes the information encapsulated in the query to obtain the exfiltrated data. Eventually, the response is sent back to the requesting host.

2.3. Encrypted DNS

As security and privacy have received more and more attention in recent years, several protocols on secure DNS have been gradually carried out. Earlier efforts include DNSSEC [23] and DNSCrypt [24]. DNSSEC adds a signature mechanism to the DNS to protect the integrity of the DNS, which can effectively prevent the tampering of DNS by the man-in-the-middle. However, DNSSEC does not provide confidentiality, that is, the DNS query is still in cleartext. DNSCrypt is implemented by the open-source community while providing integrity and confidentiality. However, it did not submit an RFC for standardization, and applications supporting DNSCrypt are very limited. In 2016, DNS over TLS (DoT) [25] was officially approved. DoT encrypts and wraps DNS queries and responses via the Transport Layer Security (TLS) protocol. Specifically, the client establishes a TLS session with a recursive resolver on port 853 and transfer DNS queries and responses over the encrypted connection. In October 2018, DNS over HTTPS (DoH) was standardized in RFC 8484 [11]. DoH encapsulates DNS queries and responses into the body of HTTP requests. The client establishes a TLS session with a recursive resolver, and HTTP requests are transferred through HTTP over TLS (HTTPS) protocol. The structure of such a DoH encapsulation is shown in Fig. 1. Compared with DoT, DoH allows

Fig. 1. The encapsulation process of DoH: A DNS query or response is encapsulated into HTTP and encrypted by TLS.

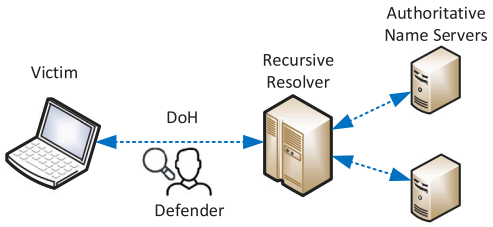


Fig. 2. Threat model.

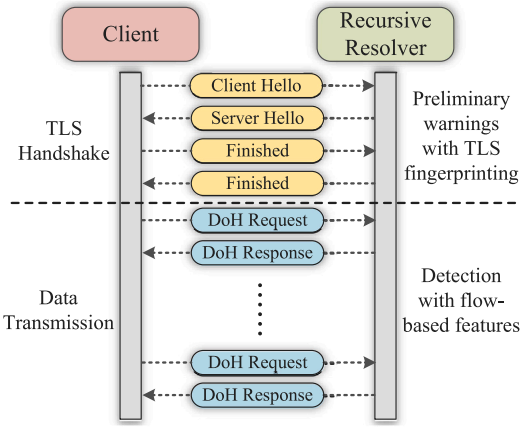


Fig. 3. The structure of the DoH flow and two detection steps of the proposed method. The TLS handshake process in the figure is a simplified illustration based on TLS 1.2.

DNS to be integrated into the HTTP ecosystem, thus the programming interface of DoH is simpler. Utilizing the features of HTTP/2, such as the server push mechanism, DoH enables the server to preemptively push the DNS responses that the client needs later, thus reducing latency.

Currently, public DNS resolvers such as Cloudflare, Google, and Quad9 provide both DoH and DoT. On the client-side, browsers such as Chrome, Firefox, and Microsoft Edge use DoH instead of DoT as the encryption scheme for DNS. After experimental support in 2019, Firefox starts using DoH as the default setting in the United States in 2020 [26]. Chromium-based browsers, such as Chrome and Microsoft Edge started rolling out DoH as an official feature in the stable release in 2020 (version 83 [27] and version 86 [28] respectively). Microsoft also announced that it will provide DoH support in Windows in the future [29]. We believe that DoH is the mainstream choice for encrypted DNS, and we focus on DoH in this work.

2.4. HTTPS interception

For HTTPS, one of the ideas for security inspection is to try to maintain visibility to traffic. To do this, the network middleware is generally used as a transparent proxy. The middleware terminates and decrypts the TLS session initiated by the client, analyzes the inner HTTP plain text, and then initiates a new TLS connection to the destination website. To decrypt, the CA certificate of the middleware needs to be deployed to the monitored device. Although this solution is effective, it is obviously expensive to deploy and causes potential privacy issues. In addition, replacing the client's certificate may cause new security issues. Recent work has pointed out that nearly all HTTPS interception software reduces connection security and many introduce severe vulnerabilities [30]. Therefore, we only focus on unencrypted features from the DoH traffic to perform data exfiltration detection in this work.

2.5. Data exfiltration of DNS over HTTPS

Since third parties have no visibility to DoH traffic, hiding the tunnel in DoH will greatly increase the possibility of attackers evading detection. Currently, the proof-of-concept of DoH tunneling has been released [31,32]. It was reported recently [18] that APT34 becomes the first known hacker group to weaponize DoH based on [32].

3. Threat model and overview of the proposed method

3.1. Thread model

We consider a scenario, illustrated in Fig. 2, in which an attacker has managed to compromise a client and use DoH tunneling for data exfiltration. We intend to detect the DoH tunneling based on the DoH traffic between the victim host and the recursive resolver. In our threat model, we do not consider decrypting DoH traffic and assume that DoH traffic and non-encrypted DNS traffic can already be distinguished. We also assume that DoH traffic and non-DoH HTTPS (standard HTTPS) traffic can already be classified. Approaches to classifying DoH traffic from HTTPS traffic have been proposed in recent works [12]. In fact, it is even possible to determine whether it is DoH traffic by simply observing the IP address of a packet and the Server Name Indication (SNI) field in the TLS handshake field (for example, if the destination IP address of a packet is 8.8.8.8 and its SNI field is dns.google, it should be a DoH packet) [33]. Therefore, we focus on detecting DoH tunneling from DoH traffic, and distinguishing DoH traffic from non-encrypted DNS traffic or non-DoH HTTPS traffic is beyond the scope of this paper. We assume that the attacker can freely change the packet length of the DNS query within the scope allowed by the DNS protocol and the attacker can also freely set the packet sending rate of the DNS query. We also assume that the attacker does not compromise other applications on the victim's host, such as compromising the browser.

3.2. Overview of the proposed detection method

The structure of the DoH flow and the proposed detection method is illustrated in Fig. 3. In general, a DoH flow can be divided into two phases. In the first phase, the client and recursive resolver exchange keys to establish a connection, and this process is called the TLS handshake. In the second phase, the client and recursive resolver start to send DNS queries/responses to each other. The proposed detection method is inspired by such a DoH flow structure, which includes two detection steps corresponding to the two phases of the DoH flow. The first step is based on the fingerprint of the TLS handshake, which can be performed as soon as the client initiates a connection, thus providing a preliminary warning of potential DoH tunneling. The second step is to extract the flow-based features of the data transmission stage and input these features into a trained classifier to detect DoH tunneling.

Specifically, for the first detection step, the plaintext information during the TLS handshake can be used for TLS fingerprinting. We measure the TLS fingerprints of the clients that currently support DoH and find that their TLS fingerprints are all unique. This means that

if an attacker initiates a tunneling connection without imitating a common TLS implementation (such as a browser), then the defender can observe its different unique TLS fingerprint. Although this is not an accurate detection, it can provide a preliminary warning, especially the TLS fingerprinting that can be performed during the handshake phase (before data transmission).

After TLS handshake and establishing an HTTPS connection, the client and recursive resolver start to send DNS queries/responses to each other. We extract features of the DoH flow in the data transmission stage and build classifiers. Since the traffic is encrypted, we can only observe features such as packet size and timestamp. We analyze and select features related to the DoH tunneling, and prove through experiments that classifiers trained with these flow-based features are effective to perform detection.

4. TLS fingerprint-based detection

When establishing a TLS connection, the client and server first negotiate the parameters for establishing the connection, including the supported cipher suites, certificate, etc., and then generate the session key to encrypt the communication data. This process is called "TLS handshake". The first step of the handshake is that the client sends a ClientHello message to the server, which is sent in cleartext. The ClientHello message provides the server with a list of cipher suites and extensions that the client supports. The cipher suite list is ordered by preference of the client, and each cipher suite defines a set of cryptographic algorithms, e.g., AES-GCM-256. The extensions provide additional information to the server that facilitates extended functionality, e.g., Server Name Indication is an extension by which a client indicates which hostname it is trying to connect to. The diversity of cipher suites and extensions makes TLS have a large parameter space, and different programs often support different cipher suites and extensions. This allows TLS fingerprinting based on the ClientHello message, which has been examined in the academic literature and utilized by the open-source community.

Some recent works [20,21,34] have discussed using TLS fingerprinting for malware detection or censorship. The basic idea of malware detection or censorship based on TLS fingerprinting is straightforward. A database is constructed by collecting TLS fingerprints, then the observed fingerprints can be compared with the fingerprints in the database. The authors of [20] pointed out that although malware detection based only on TLS fingerprinting is not effective enough, TLS fingerprinting provides valuable information that can be combined with other indicators for further detection. Inspired by this conclusion, we believe that TLS fingerprinting of a DoH connection may be helpful for DoH tunneling detection.

However, to the best of our knowledge, there is no TLS fingerprinting measurement for DoH clients. We perform extensive TLS fingerprinting measurement on clients that currently supports DoH to fill this gap. In general, we select three types of applications: browser, DoH proxy, and proof-of-concept for DoH tunneling. For the browser, we chose Chrome and Firefox, which are the most representative applications that support DoH. Various DoH proxy software has been developed since applications other than browsers generally do not yet support DoH. The DoH proxy listens and intercepts local non-encrypted DNS queries, and encapsulates non-encrypted DNS queries to DoH. We choose open source DoH proxies that remain active as of this writing, some of which are developed by commercial companies and others by the community. The proof-of-concept of DoH tunneling implements data exfiltration over DoH. We use TLS fingerprinting method provided in [34] for our measurement since it provides the largest real-time updated fingerprint database to our knowledge. We use each DoH client to initiate 10 sessions through three public recursive resolvers (Cloudflare, Google, Quad9) and record fingerprints. The detailed information of these applications and the fingerprinting results are listed in Table 1.

The results show that the measured DoH clients all have different TLS fingerprints. Previous studies have pointed out that client-side support for cipher suites and extensions is related to operating systems and programming languages (since TLS-related libraries are usually called during the development process) [34]. The fingerprinting result of DoH clients conforms to this trend. Clients implemented by different programming languages have different fingerprints. The same client may have different fingerprints when running on different operating systems. Different DoH clients written in the same programming language may also have different fingerprints due to the diversity of implementation methods.

The uniqueness of the DoH client's fingerprint provides a simple but efficient preliminary method for DoH tunneling detection, establishing a whitelist mechanism based on TLS fingerprinting. This is because, unlike the common HTTPS, DoH connections are generated by very few applications (since DoH is not natively supported by most applications at present). The vast majority of DoH connections may be generated by the browser and a small number by the DoH proxy. After adding the fingerprints of the browser and DoH proxy to the whitelist, a connection is more likely to be malicious if its fingerprint does not match the whitelist.

Fingerprints cannot directly indicate the existence of DoH tunneling, but provide defenders with directions for further investigation. In particular, fingerprinting is based on the first packet of DoH, which means an early warning is possible before data transmission. In practice, the fingerprinting database needs to be collected and updated. Since the number of applications that support DoH is much smaller than those that support TLS, the cost of maintaining the DoH fingerprinting database is relatively low, and the existing TLS fingerprinting-related toolchain can be reused.

To evade detection based on TLS fingerprinting, attackers need to mimic popular TLS implementations such as browsers. Recent work [34] has pointed out that TLS implementations can be difficult to mimic correctly due to different TLS functions supported by different libraries. Besides, it is burdensome for attackers to know what fingerprints would be good candidates to mimic since the fingerprints of benign applications (such as browsers) are changing with version updates. In summary, although detection based only on TLS fingerprinting may not be effective enough, it can provide administrators with an effective preliminary warning, and greatly increases the difficulty of a successful attack.

5. Flow-based detection

In this section, we introduce the proposed flow-based detection. The basic idea of detection is straightforward: extracting features from DoH tracing and training classifier, and using the trained classifier for detection. Following this strategy, we collect a large number of DoH traffic (including benign and tunneling traffic) to train and test models. In Section 5.1, we describe the details of data collection. In Section 5.2, we elaborate on which features are extracted. Finally, how classifier models are selected is presented in Section 5.3.

5.1. Data collection

To collect benign and tunneling DoH traffic, we adopt various settings to simulate possible environmental differences between the victim and attacker. This is because the environment between the victim and attacker is varied in practice, which may affect the detection. For example, if the attacker's server is geographically far away from the victim, the DNS query and response of the tunnel may have a high latency compared to the benign DNS since benign DNS queries are often directly responded to by the recursive server due to cache, while DNS queries of the tunnel must be sent to the attacker's server. Therefore, such latency can be used as a feature to distinguish whether the DoH traffic is malicious. Specifically, the settings we adopt are shown in

Table 1
TLS fingerprints for DoH clients.

Type	Name	Version	Written in	OS	TLS Fingerprint[g]
Browser	Chrome	86	Mostly C/C++/Rust	Windows macOS	9c673fd64a32c8dc 9c673fd64a32c8dc
	Firefox	82	Mostly C/C++	Windows macOS Linux	4d578e2d2851f526 4d578e2d2851f526 4d578e2d2851f526
DoH Proxy	cloudflared[a]	2020.10.2	Go	Windows macOS Linux	9cd6b6b8a763c894 9cd6b6b8a763c894 9cd6b6b8a763c894
	DNSproxy[b]	0.33.1	Go	Windows macOS Linux	c9340c63cd64e4e1 c9340c63cd64e4e1 c9340c63cd64e4e1
	doh-proxy[c]	0.09	Python	Windows Linux	70ed3f034ddc6f77 cbe3ec81604fb67e
	dnscrypt[d]	2.0.44	Go	Windows macOS Linux	a91c0644c199823d a91c0644c199823d a91c0644c199823d
	https-dns-proxy[e]	[f]	C	macOS Linux	37695dd988f0c8b8 37695dd988f0c8b8
PoC for DoH tunneling	godoh [31]	82	Go	Windows macOS Linux	b9b6dcd8542d8ad9 b9b6dcd8542d8ad9 b9b6dcd8542d8ad9
	DNSExfiltrator [32]	86	C#	Windows	d00cb24ff887292e

[a] https://github.com/cloudflare/cloudflared
[b] https://github.com/AdguardTeam/dnsproxy
[c] https://github.com/facebookexperimental/doh-proxy
[d] https://github.com/DNSCrypt/dnscrypt-proxy
[e] https://github.com/aarond10/https_dns_proxy
[f] No releases, version committed on Sep 25, 2020 is used.
[g] Details of fingerprints can be viewed in tlsfingerprint.io. e.g., Chrome 86: https://tlsfingerprint.io/id/9c673fd64a32c8dc

Table 2
Overview of the settings of our dataset. The dataset includes benign DoH traces and DoH tunneling traces and we take different settings to collect traces.

Settings	Number of settings	
	Tunneling	Benign
Location	2	2
Recursive Resolver	3	3
Time Interval	7	–
Packet Length	227	–
Website	–	1500

Table 2. For benign and tunneling traffic, we use 2 different locations and 3 different resolvers. For benign traffic, we query the DNS records of 1500 websites through DoH. For tunneling traffic, we use 7 different time intervals and 227 packet lengths to simulate the different settings that an attacker might use to exfiltrate data. Below we describe in detail the process of collecting traffic.

We set up three virtual machines running Ubuntu 20.04 as the operating system. Two virtual machines are located in Seattle and the other in Singapore. One virtual machine in Singapore and one in Seattle simulate the victim. One virtual machine located in Seattle simulates being controlled by the attacker. All virtual machines are configured with public IP addresses and can communicate with each other. Regarding the recursive resolvers, we choose Cloudflare, Google, and Quad9, which are well-known public resolvers. We capture network traffic between the virtual machine and the recursive resolver using *tcpdump* and filter the traffic by IP address and port to obtain the final DoH traffic trace. We only focus on TLS packets with application data and filter insignificant packets with no payload such as TCP ACK packets. For benign DoH traffic, we adopt a data collection method similar to [14]. Two virtual machines that simulate victims automatically visit the top 1500 websites in Alexa Top Sites and send DNS requests through three different recursive resolvers. We collect DoH traffic for each visit. Specifically, each website visit is executed by Firefox 82 (we use Firefox because Chrome does not support DoH on Linux at the time of this writing). Firefox is set to use DoH only (by setting the Firefox parameter `network.trr.mode` to 3) to avoid possible non-encrypted DNS affecting data collection. Selenium (version 3.141.0) is used to drive Firefox to automatically visit a website in headless mode, under which Firefox runs without showing a GUI. Scheitle et al. [35] point out that there is a possibility that malicious websites may rank high in Alexa through a burst of requests from a high number of infected clients. It is worth mentioning that for our process of collecting benign DoH traffic, the DoH traffic during Firefox's visit to websites only contains DNS queries and responses and not DNS tunneling traffic, regardless of whether the visited website itself is benign or not. In fact, we check through Virustotal that the URLs we used are benign. Therefore, the collected DoH traffic is benign.

To configure the DoH tunneling, we register a domain with Godaddy, use the virtual machine in Seattle as the Name Server (NS) for that domain, and run the server-side program of DNSExfiltrator [32] on this virtual machine (i.e., this virtual machine is simulated as being controlled by an attacker). The other two virtual machines are simulated as victims, running the client-side program of DNSExfiltrator, and using three different recursive resolvers to send local files (we use text files) to the server controlled by the attacker through the DoH tunneling. Each time we perform data exfiltration through DoH tunneling, we set different time intervals of DNS queries or different full domain name lengths. We perform each set three times and collect traffic traces. Specifically, the time interval between two queries is random. The overall average time interval of a DoH flow is about 1, 5, 10, 20, 100, 500, 1000 ms (A total of 7 time interval settings). The length of the full domain name ranges from 27 to 253 (A total of 227 packet length settings). Each DoH tunneling flow contains at least 5 sets of DNS queries and responses.

In summary, we consider different scenarios varying recursive resolver, geo-location, packet size, and packet sending rate. We designed

Table 3
Features.

Category	Feature	Category	Feature
TLS record length of queries	src2dst_min_length src2dst_max_length src2dst_mean_length src2dst_stddev_length	time intervals between two consecutive queries	src2dst_min_time src2dst_max_time src2dst_mean_time src2dst_stddev_time
TLS record length of responses	dst2src_min_length dst2src_max_length dst2src_mean_length dst2src_stddev_length	time intervals between two consecutive responses	dst2src_min_time dst2src_max_time dst2src_mean_time dst2src_stddev_time
TLS record length of all packets on a flow	bidirectional_min_length bidirectional_max_length bidirectional_mean_length bidirectional_stddev_length	time intervals between an adjacent pair of a query and response	bidirectional_min_time bidirectional_max_time bidirectional_mean_time bidirectional_stddev_time

heterogeneous experiments based on the collected data to discuss the influence of these factors on DoH tunneling detection.

5.2. Features

A DoH traffic trace can be considered as a time sequence of packets. Due to the encryption performed by DoH, only the size, timestamp and direction are useful for detection (the destination port, usually 443, is a constant, and the destination IP address belongs to the recursive resolver). Previous works [13,14] usually use a high-level representation of the network (Netflow) to extract statistical features of encrypted traffic. For the scenario we consider, a DoH flow can be defined as a traffic stream transmitted via HTTPS between a client and a recursive server over a specific port. Specifically, we denote the client as the source and the recursive server as the destination. Packets sent from the source to the destination are DNS queries, and vice versa are DNS responses. We reassemble the DoH traffic trace to flows and extract flow-level statistical features based on the packet size, timestamp, and direction in each flow.

In particular, we calculate the *minimum, maximum, mean, standard deviation* of each flow-level feature. The feature values are decimal numbers, which can be directly consumed by machine-learning models. We are inspired by previous works on DNS tunneling and encrypted traffic and combined with the characteristics of DoH to select features. There are also some trade-offs (we do not use some features from previous works). All features we used are listed in Table 3 and below we give details and theoretical analysis of each feature. Through the experiments (see Section 6), we demonstrate that the features selected in this paper are sufficient for DoH tunneling detection.

5.2.1. TLS record length

The length of the query domain name has been used as an important feature in previous works to detect DNS tunneling [3,6,36]. This is because the data exfiltration through DNS essentially encodes the data into a subdomain string, and constructs the queried domain name with that string and a high-level domain name. The tunneling application makes a request for resolving that domain name, and then NS controlled by the attacker will receive the request and get the information by decoding the subdomain string. Therefore, a longer domain name helps to transmit more data.

However, the length of the query domain name cannot be directly observed from the encrypted DNS. We focus on the TLS record of a DoH packet, whose length can indirectly indicate the length of the domain name to some extent. Since DNS queries/responses have length limits (the total length of a domain name cannot exceed 255 bytes), they fit in one single TLS record in most cases even when they are wrapped in HTTP requests rather than being split into multiple TLS packets [14]. Cipher suites in TLS include stream ciphers (e.g. `ChaCha20`) and block ciphers (e.g. `AES`). For stream ciphers, they encrypt data byte by byte and the encrypted length is equal to the plain data length. For block ciphers, they encrypt a block of data of a fixed length and the end of the data may need some padding to make the length an integer multiple of the block. Although the length of the ciphertext and the plaintext may not be equal, the length of the two is still positively correlated. In general, when the DNS query does not include padding, the length of the domain name can be indirectly observed through the TLS record length. We consider the TLS record length of DoH queries and responses. Specifically, we consider the TLS record length of queries (*src2dst*), the TLS record length of responses (*dst2src*), and the TLS record length of all packets on a flow (*bidirectional*).

5.2.2. Time intervals

Time intervals between packets are common features for detecting DNS tunneling [4,19,37]. Recursive resolvers usually cache the information about previous DNS lookups to reduce the latency. When a DNS response includes a time-to-live (TTL) value, the receiving recursive resolver will cache that record for the time specified by the TTL. Then if the recursive resolver receives a query before the expiration time, it simply replies with the already cached record rather than retrieves from the Authoritative DNS server again. In contrast, the recursive resolver cannot find the domain name generated by DNS tunneling in the cache. Each query received is forwarded to the NS controlled by the attacker. The corresponding response will be returned from the NS instead of directly from the cache of the recursive resolver. Therefore, the time interval between an adjacent pair of a query and response of tunneling traffic is often longer than that of normal DNS traffic. In addition, normal DNS traffic is often random, while DNS requests are often sent at a relatively constant rate when transferring data through DNS.

Fortunately, the extraction of time-related features is not affected by encryption. In general, we consider the time intervals both between an adjacent pair of a query and response and vice versa (*bidirectional*), and between two consecutive queries (*src2dst*), and two consecutive responses (*dst2src*).

5.2.3. Some features that we discarded

DoH flows generated by different queries (e.g., DoH queries generated by accessing different websites) may have different cumulative bytes or durations. Intuitively it is possible to distinguish traffic based on the cumulative bytes or duration of the flow. Previous works related to encrypted DNS traffic used such features. For example, the authors of [12] use the duration of the flow as one of the features to distinguish DoH traffic from non-encrypted DNS traffic. The authors of [13] use the total number and cumulative bytes of queries and responses of a flow as features for website fingerprint.

For a DoH tunnel used for data exfiltration, it will theoretically generate more bytes than normal traffic. Does this mean that the cumulative bytes and duration of the flow can be used as features to detect DoH tunneling? The answer is no. This is because previous work using these features contains the implicit assumption that a DoH flow corresponds to a query/response generated by a user's action (e.g., visiting a website at a time), and thus different DoH flows have different cumulative features. However, a more practical situation is

that the browser may visit multiple websites at the same time. At this time, the generated queries/responses may be in a single DoH flow [14], which means that a normal DoH flow may also have larger cumulative bytes or duration. Therefore, we do not use such features in this work.

5.3. Model selection

Unlike some work in other fields to extract high-dimensional features, the number of features that can be extracted from DoH traffic is limited due to encryption. Therefore, we do not build complex classifiers or even neural networks. On the contrary, we believe if a simple classifier is effective, it can lead to acceptable engineering costs. In addition, the interpretability of neural networks may be insufficient. While in the field of cyber-security, interpretability can help defenders analyze security incidents and adopt more appropriate strategies.

We select three interpretable supervised models: Boosted Decision Tree (DT), Random Forest (RF), and Logistic Regression (LR). We conduct a lot of analysis based on the interpretability of the model in the experiment. In particular, our experiments include adversarial considerations and demonstrate that the interpretability of the model can help prioritize DoH flows when the attacker evades detection, thus providing the defender with useful information for attack investigation. We do not choose a specific model but adopted these three models in all experiments. From the preliminary results, we found that RF and DT perform better in most cases. However, we find that LR, as a linear model, has unique advantages for prioritizing DoH flows based on the scores of LR. More details are discussed in Section 6. We implemented these models with scikit-learn [38], leaving the default parameters unchanged.

In addition, compared with an anomaly detection model that only uses benign traffic for training, the supervised learning method that uses both benign and malicious traffic training models usually has higher accuracy, since the model can directly fit the decision boundary of the two types of data. But obviously, the required labeled malicious traffic data increases the difficulty and workload of the model training. The malicious traffic we collect is simple, generated by open-source tools with basic configuration (described in Section 5.1). In the next section, we evaluate the effectiveness of the trained model for more advanced attackers trying to evade detection. We demonstrate through experiments that the models trained on benign traffic and simple malicious traffic can remain effective against advanced attackers in most cases (some subtle adjustments to the model are required at most). This means that only simple, easy-to-collect malicious traffic needs to be used to train the model in our method, instead of collecting malicious traffic that advanced attackers might use. Therefore, the difficulty of data collection is greatly reduced, and the practicability of our method is improved.

6. Evaluation on flow-based detection

6.1. Evaluation setup

For the evaluation of flow-based detection, we divide the dataset into a training set and a test set on a 4-to −1 basis. We train the model with 5-fold cross-validation and select the model that achieves the highest F1-score on the validation set at a threshold of 0.5 for testing. The metrics used for testing include accuracy, precision, recall, and F1-score. Specifically, when a flow is labeled as benign, it is called a *positive*. A *true-positive* (TP) is a benign DoH flow that is correctly labeled as benign and a *false-positive* (FP) is a DoH tunneling flow that is incorrectly labeled as benign. On the contrary, a *true-negative* (TN) is a DoH tunneling flow that is correctly labeled as malicious and a *false-negative* (FN) is a benign DoH flow that is incorrectly labeled as malicious. Then the metrics we use can be defined as:

$$\begin{aligned}\text{Accuracy} &= \frac{TP+TN}{TP+FN+FP+TN},\\ \text{Precision} &= \frac{TP}{TP+FP},\\ \text{Recall} &= \frac{TP}{TP+FN},\\ F_1\text{-score} &= 2\cdot\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}.\end{aligned} \tag{1}$$

Our evaluation considers different scenarios varying locations, DoH recursive resolvers, padding, etc., to evaluate the impact of these factors on model performance. To this end, we design a series of heterogeneous experiments. In each experiment, we vary a factor (e.g., location or recursive resolver) to evaluate the performance of classifiers and discuss possible ways for attackers to evade detection. Specifically, we first use all the training data and features to train and test the model, which is an ideal scenario and shows the *upper bound* of the model performance (Section 6.2). Then we test the influence of location (Section 6.3), DoH Recursive Resolver (Section 6.4), padding (Section 6.5), and time interval (Section 6.6) on the performance of the model. Moreover, we simulate a scenario where an attacker may evade detection (Section 6.7). We found that it is difficult (or even only theoretically possible) for an attacker to achieve such evasion, and the transmission rate of the tunnel is extremely low at this time. More importantly, our method can still help administrators investigate the security status more quickly in this case. Finally, we summarize the experimental results from the perspective of both defenders and attackers (Section 6.8).

Table 4
Experimental results of the overall dataset.

Classifier	Accuracy	Precision	Recall	F1-score
DT	0.999	0.999	0.999	0.999
RF	0.999	0.999	0.999	0.999
LR	0.978	0.987	0.968	0.978

6.2. Evaluation of the overall dataset

We first conduct a straightforward experiment to preliminarily test the performance of the classifier. We combine all the collected data into an overall dataset as described in Section 5.1. All the features (a total of 24 features, see Section 5.2) are used by the classifier for training and testing. This experimental setup means that we do not consider the possible effects of different locations and recursive servers and assume that the attacker does not take measures to evade detection.

The experimental results are summarized in Table 4. LR has some false positives, so its F1-score is relatively low at 0.978. Although this experimental setup is carried out in an idealized scenario as mentioned above, the experimental results can demonstrate that the DoH tunneling traffic is different from the normal DoH traffic, and the extracted features can express such a difference to build an efficient classifier.

The classifiers, especially RF and DT, perform so well in this experiment. Does it mean that DoH tunneling can be effectively detected? The answer is of course not. In fact, an attacker can easily change some feature values of a flow, such as the packet sending rate. In theory, the attacker may evade detection. We need to consider the robustness of the detection method when facing a practical adversary. Therefore, in addition to the effectiveness of the model itself, we are more concerned about how attackers can evade detection. In other words, we hope to find the *lower bound* of the detection ability of the proposed method, not just the performance of the model on an idealized dataset. If the attacker needs to pay an extremely high cost (even only theoretically feasible) to evade the proposed method, the proposed method is effective for DoH tunneling detection.

Table 5
Performance variation changes in location (F1-score, standard deviations less than 2%).

Train	Test		
	Model	Seattle	Singapore
Seattle	DT	0.999	0.984
	RF	0.999	0.996
	LR	0.974	0.968
Singapore	DT	0.759	0.999
	RF	0.951	0.999
	LR	0.726	0.988

In subsequent experiments, we analyze the proposed method from the perspective of attackers and defenders. We vary one factor (e.g., location or recursive resolver) in each experiment to evaluate the performance of classifiers and discuss possible ways for attackers to evade detection.

6.3. Influence of locations

DoH traces may vary across locations. For benign DNS queries, this is because websites may adapt their content to specific geographic regions, and the resources cached by resolvers may vary across different regions. For DoH tunneling, the query through the tunnel will inevitably reach the NS controlled by the attacker. So intuitively, the time required to transmit data through the tunnel to the same NS at different locations is different.

In this experiment, cross-validation is carried out through datasets collected at different locations. The victims are located in Singapore and Seattle, and the NS controlled by the attacker is located in Seattle (see Section 5.1). The benign DoH tracing and DoH tunneling tracing collected in Singapore form one dataset, while another in Seattle. The two datasets are validated as training sets and test sets respectively. Table 5 shows the classifier performance when crossing these datasets for training and testing.

When trained and tested on the same location, the results of the classifiers are similar to those on the overall dataset, and some F1 scores even increased by about 0.1%. When trained and tested in different locations, the F1-score is reduced by 4% to 26%. The decline occurred mainly in training on the dataset collected in Singapore and testing on the dataset collected in Seattle. RF performs better than DT and LR.

Specifically, we find that the main reason for the low F1-score is a large number of false positives, but not many false negatives. For example, the precision is 0.585 and the recall is 0.958 when the F1-score of the LR is 0.726. The precision is 0.611 and the recall is 0.999 when the F1-score of the DT is 0.759.

We believe the reason for this result is that the time-related features have a higher importance in the classifier, and the time-related features of DoH traces generated by different geographic locations are different. In our experimental setup, it takes a long time for data packets to be transmitted from the victim in Singapore to the NS in Seattle, while it only takes a short time in Seattle. Therefore, for the classifier trained based on the data collected in Singapore, its decision boundary for time-related features is not completely applicable to the data collected in Seattle. In turn, the classifier trained on the data collected in Seattle has stricter decision boundaries, so it still performs well on the data collected in Singapore. For benign DoH traffic, we believe that resolvers usually cache popular resources, and resolvers and CDNs use geographic location methods to load-balance requests. Therefore, benign DoH traces do not change much at different locations, and the recall of cross-validation does not significantly decrease. As for the performance of models, we believe that the reason why the RF can maintain a relatively good effect is that its majority voting mechanism reduces the impact of time-related features, and the TLS record length-related features are helpful for detection.

Table 6
Performance variation changes in recursive resolvers (F1-score performed with RF by the host in Seattle).

Train	Test		
	Cloudflare	Google	Quad9
Cloudflare	1	0.984	0.963
Google	0.661	1	0.667
Quad9	0.752	0.662	0.999

According to the results of this experiment, for the defender, it is necessary to use a locally trained classifier to ensure effectiveness. For the attacker, it is necessary to make the NS as geographically close to the victim as possible, thus reducing the latency of the DoH tunneling from the victim to the NS. However, geographic proximity alone is not enough to evade detection (high F1-score when training and testing the model in the same location).

6.4. Influence of DoH recursive resolver

We collect DoH traces based on three public recursive resolves, Cloudflare, Google, Quad9. These data are used as training set and testing set for verification respectively. Some of the experimental results (F1-score performed with RF by the host in Seattle) are shown in Table 6 due to space limitations. Results based on DT and LR are similar (within 5%) to those in Table 6. The results for the host in Singapore are also similar (within 3%) to those in Table 6.

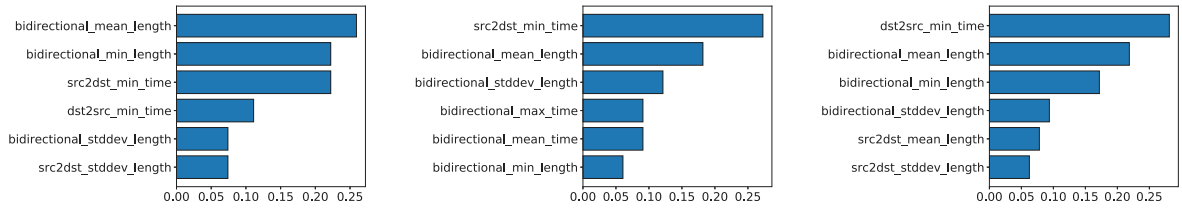
In general, training and testing on the same resolver yield the best results as expected, and performance decrease in other cases. Specifically, the model can maintain relatively good results on the Google and Quad9 datasets when training on the Cloudflare dataset. When training on Google or Quad9 datasets, the F1-score of the model on other datasets is reduced by 25% to 44%

To investigate this asymmetric decrease we rank the features according to their importance for the classifier. Normalized feature importance of RF trained with different resolvers datasets is shown in Fig. 4 (only the top-6 features are shown due to space limitations). We find that the important features in the Cloudflare dataset are also important in the Google dataset and the Quad9 dataset (e.g., *bidirectional_mean_length*). While some important features in the Google dataset or Quad9 dataset are not so important in other datasets (e.g., *dst2src_min_time* is the most important feature in the Quad9 dataset, but it is not the top-6 important features in the Google datasets). This difference may be caused by different implementations and deployment strategies of different resolvers, such as the difference in response time caused by different caching strategies when resolving domain names. The difference in feature importance could induce erroneous splits early in the trees causing the performance drop. In conclusion, the defender should collect data from different recursive resolvers for training according to these experimental results.

6.5. Influence of EDNS(0) padding

The above experiments simulate an attacker trying to change the geographic location of controlled NS and recursive resolver to evade detection, and the results demonstrate that such changes are not sufficient to evade detection. From the perspective of the attacker, it is necessary to find possible evading ways from the features of the classifier. In general, the features of the classifier are composed of two types: TLS record length-related features and time-related features. In this experiment, we evaluate the influence of TLS record length on detection.

Obviously, the basic idea to evade detection is to make the TLS record length similar to that of the benign DoH packet. If an attacker only adjusts the length of the domain and the length of the encoded information, it is not easy to determine an optimal length value to



(a) Cloudflare (b) Google (c) Quad9

Fig. 4. Normalized feature importance of random forest trained with different resolvers datasets by the host in Seattle.

Table 7
Performance of classifies based only on time-related features.

	Model	Precision	Recall	F1-score
Overall	DT	0.989	0.984	0.986
	RF	0.992	0.991	0.991
	LR	0.91	0.888	0.899
Seattle	DT	0.991	0.991	0.991
	RF	0.994	0.996	0.995
	LR	0.894	0.959	0.925
Singapore	DT	0.992	0.991	0.991
	RF	0.998	0.995	0.996
	LR	0.748	0.995	0.854

Table 8
The average predicted probability of each classifier for DoH tunneling flows with different sending intervals and the F1-score obtained by each classifier.

Interval	Probability			F1-score		
	RF	DT	LR	RF	DT	LR
2 s	0.94	0.745	0.658	0.99	0.99	0.92
4 s	0.92	0.745	0.472	0.99	0.99	0.67
8 s	0.89	0.745	0.297	0.99	0.99	0.62
20 s	0.89	0.745	0.099	0.99	0.99	0.59

evade detection. As described before, we hope to find the *lower bound* of the detection ability of the proposed method. Therefore, we assume a scenario that is most beneficial to the attacker, both benign DoH traffic and DoH tunneling traffic are perfectly padded with EDNS(0) padding option.

Before describing the experiment, we first introduce the background of EDNS(0) padding. Extension mechanisms for DNS (EDNS) is a specification to increase the functionality of DNS [39]. The EDNS(0) padding option is published in [40] and padding policies are described in [41]. Padding is used to pad DNS queries and responses to a desired size, which makes it more difficult to analyze traffic based on packet length. Padding is expected to be used with DoH together to better protect privacy. All DoH data packets are of fixed length under a strict padding policy. Therefore, all TLS record length-related features are invalid in our hypothetical scenario. In this experiment, we evaluate the performance of classifiers based only on time-related features. Classifiers are trained and tested with the overall dataset, Seattle and Singapore dataset respectively.

The experimental results are summarized in Table 7. Compared with previous experiments (Tables 4 and 5), the performance of all classifiers is decreased. The F1-score of the LR model drops up to 13%. In contrast, RF and DT have a small decrease in performance with about a 1% drop in F1-score.

This experimental result shows that although the features related to the TLS record length do affect the detection results, the classifier can still maintain acceptable performance. In addition, recent work has pointed out that there are not many clients that currently support padding, and the implementation of padding is not standardized enough [14]. This means that in the current real world, an attacker is unlikely to have the ideal conditions we assume. In conclusion, the defender should train classifiers based only on time-related features to detect potential padding-based evasion, while it is difficult for the attacker to evade detection only by modifying the TLS record length.

6.6. Influence of time interval

Now the attackers can only find possible ways to evade detection based on time-related features according to the above experimental results. In this experiment, we evaluate the influence of time-related features on detection. We assume that the attacker has used the EDNS(0) padding option described in the previous experiment, and on this basis, the attacker further adjusts the time-related features of DoH tunneling to evade detection. The reason for making this assumption is that although padding alone is not enough to evade detection, it does degrade the performance of the classifier. Therefore, modifying time-related features while using padding is the most beneficial scenario for the attacker.

Intuitively, when the data transmission rate of the DoH tunneling is slow, its camouflage may be better. The largest DNS query sending interval in the DoH tunneling dataset we have used so far is 1 s (see Section 5.1). In this experiment, we further collected DoH tunneling traffic with time intervals of 2, 4, 8, and 20 s (same with Section 5.1, these are average time intervals on a flow, the time interval between two queries is random). Traffic is collected on the host in Seattle. Each DoH flow contains at least 5 sets of DNS queries/responses.

For the experiment, we first use the previously trained model to detect the newly collected data. The average predicted probability of each classifier for DoH tunneling flows with different sending intervals is shown in Table 8. As a result, RF and DT still accurately detect DoH tunneling with all sending rates (F1-score>0.99), while LR only detected DoH tunneling with a sending interval of 2 s (F1-score is about 0.92 when the interval is 2s and the DoH tunneling flow in the remaining intervals is classified as benign). In contrast, the predicted probability of RF and DT for DoH tunneling flows is often more than 0.9 in previous experiments. Therefore, although increasing the time interval for sending DNS queries does not cause false positives for these classifiers, it does help to evade detection. The predicted probability of DT is consistent. This indicates that when the DT detects flows with different sending intervals, it may have reached the same internal node (i.e., a feature and decision condition), and then the flow is classified as DoH tunneling. This further shows that the attacker needs to adjust other features to evade detection. For LR, as a linear model, the increase in time interval directly leads to a decrease in its predicted probability for DoH tunneling flows, which leads to false positives.

Then we use the new data along with the previous data to retrain and test the model. RF and DT achieve almost the same results as those in Table 4. LR achieve a precision of 0.909, an recall of 0.956, and an F1-score of 0.932. Compared with only detecting DoH tunneling with a sending interval of 2 s, the performance of the retrained LR model has been greatly improved. The precision of each classifier is not significantly decreased, which means that after adding the new data for training, the classifier can still find a clear decision boundary. The above results indicate that the defender should collect DoH tunneling traffic at multiple rates as much as possible to train the model. Even if the attacker uses a slower rate than the training set, the model is still likely to be effective.

6.7. Prioritizing the detected flows

Few remaining parameters can be adjusted by an attacker to evade detection. Since the DoH tunneling flow we have collected contains multiple DNS queries, we finally consider a scenario where a DoH tunneling flow contains only one DNS query. Intuitively, such a setting could invalidate some statistics of time-related features (for example, *mean, standard deviation*). The same as the previous experimental process, we use the trained model and the retrained model to test the newly collected traffic. We found none of the models can detect tunneling at this time. It seems that there is finally a way for attackers to evade detection. Since a flow contains only one DNS query/response, the transmission rate of the established DoH tunnel is very low. Specifically, assuming that the length of the subdomain is 100 (which is not a short length for DNS) and with base64 encoding, then each query can transmit about 75 bytes. The TLS handshake for each new DoH connection takes some time (depending on the network environment and computing power). Therefore, even if there is almost no interval between the two connections, the overall rate of the tunnel is only about hundreds of bytes per second.

However, we find that LR, which has been performing relatively poorly in previous experiments, can help defenders at this time. When looking at the scores of LR, we find that although the score for DoH tunneling flow is greater than 0.5 (0 for DoH tunneling, 1 for benign DoH, threshold 0.5), the scores for DoH tunneling flow are still usually lower (e.g., 0.6 to 0.7) than that for benign DoH flow. While scores for DoH tunneling by RF and DT are usually between 0.9 to 1.0, which are similar to that for benign DoH flow.

For DoH tunneling, some inherent features are difficult to imitate. For example, the query of the DoH tunneling will inevitably reach the NS controlled by the attacker, while the benign query may just arrive at the resolver since the cache. In this case, the DNS query/response time is different. Although there are fewer features and the difference in features may not be enough for the classifier to find a good decision boundary, this difference will still be reflected in the final score for linear models such as the LR model. However, the RF and DT may directly classify the DoH flow as benign at an internal node, which cannot reflect this difference.

According to the experimental results, defenders can prioritize DoH flows based on the scores of the LR model, and give priority to investigating more suspicious flows. Although the classifier cannot detect DoH tunneling accurately in this experimental setup, we believe that the investigation based on priority can greatly help defenders to confirm the security status more quickly.

6.8. Summary

We summarize the experimental results from the perspective of defenders and attackers.

For defenders, TLS fingerprints should be monitored for preliminary detection. Different types of data should be collected as many as possible to train classifiers, including traffic from different resolvers, DoH tunneling traffic with different packet sending rates, etc. If the protected hosts are located in different geographic locations, models should be trained in different geo-locations separately. In addition to paying attention to traffic that has been identified as tunneling by the classifier, the defender should also pay attention to traffic with a high anomaly score.

There are a set of threats to the detection model and mitigation methods. For attackers, to evade the proposed detection method, firstly, the controlled NS should be geographically close enough to the victim. The DoH tunneling tool delivered to the victim's host needs to mimic popular TLS implementations such as browsers. In addition, attackers need to use means such as padding to disguise the packet length. Moreover, the attacker should send packets at a slow speed, and the number of packets in a DoH flow should be small (even only a pair of DNS query and response in a flow). To achieve all the above conditions, the attacker needs to pay a great cost, and some of the conditions may even only be theoretically feasible. Even if the attacker evades detection, the transmission rate of the established DoH tunnel is very low. Therefore, the proposed method is effective for detecting DoH based data exfiltration.

7. Discussion

In this section, we discuss several observations we gained and our lessons learned from this work.

There is no doubt that encrypted DNS is a great step for privacy, but it has caused great challenges for security detection. We show that the proposed method can detect DoH tunneling based on TLS handshake information and traffic analysis. However, compared to detection on unencrypted DNS, the accuracy of the proposed method may decrease after losing visibility to the domain name. In addition, other useful indicators can be obtained from the unencrypted DNS query/response, such as the domain name and IP used by the attacker. In practice, threat intelligence can be further constituted based on these indicators of compromise. However, these indicators cannot be observed from encrypted DNS.

In our work, fingerprint-based detection is the first stage to provide defenders with early warning and preliminary screening. The second stage of the ML-classifier used in the proposed method does not rely on TLS fingerprints, and the experiments prove its effectiveness. As of this writing, Apple has just announced that it will provide system-level TLS support for the new iOS and macOS [42], which allows apps to enable system built-in DoH. This may change the ecosystem of DoH clients and make the TLS fingerprinting ineffective. But as far as we know, some OSs such as Linux currently has no plans to support DoH at the system level [43], so TLS fingerprinting is still valid in many cases. Nevertheless, TLS fingerprinting is implicated in many studies in the field of network and security (e.g., identifying IoT devices, detecting malware, etc.), and the research community should pay continuous attention to the impact of system-level TLS support on fingerprints.

We focus on the detection of data exfiltration over DoH in this work. Intuitively, there is a difference in traffic between DoH tunneling and benign DoH query/response, so the DoH tunneling can be detected through traffic analysis. We have observed in experiments that when a DoH tunneling flow contains only a set of DNS queries/responses, the proposed detection method may not be able to detect the DoH tunneling. For data exfiltration, the low transmission rate will greatly increase the difficulty for attackers to successfully transmit data. But let us take a closer look at whether it can be detected by observing the traffic between the victim and the recursive resolver, if it is a single C&C command (even just a single bash command 'ls' transmitted through a tunnel) transmitted through DoH? In fact, typical metrics based on traffic traces may be inadequate to detect C&C commands based on encrypted DNS. As far as we know, even for DNS in plain text, such detection is also very difficult. This issue needs to be further explored by researchers.

In addition, to the best of our knowledge, research works on encrypted DNS so far are usually based on datasets collected by researchers or simulated experimental results (including this work). Of course, this is because there is a lack of encrypted DNS data from the real environment. The analysis and results in the experiment may be different from the real environment. With the further development and popularization of encrypted DNS, we hope that there will be more empirical studies on encrypted DNS from the security community.

Furthermore, since the adversarial attack on machine learning has evolved rapidly in recent years, there may be many powerful ways to evade our ML-based detection method. Although our experiments include adversarial settings, we believe that the adversarial attack is still an important open research issue. Future works should further explore the impact of adversarial attacks on detection.

8. Related work

Data Exfiltration over DNS. DNS provides a highly attractive channel for attackers who want to transfer data across a network perimeter. Detecting data exfiltration over DNS has attracted much attention from researchers in the past decade. Paxson et al. [3] develop a DNS tunneling detector by calculating the information conveyed by DNS queries and responses. Liu et al. [4] use four kinds of features including time-interval features, request packet size features to train a classifier to detect DNS tunneling. Liu et al. [5] propose a deep learning method called Byte-level CNN to detect the DNS tunneling, which can make full use of the overall information inside DNS packets. Luo et al. [44] present a DNS tunneling detection method that can detect tunneling based on A and AAAA resource records. Ahmed et al. [19] have drawn insights into DNS queries for detecting data exfiltration by their anomaly scores, the trace of query count over time, enterprise hosts querying them, and TTL and Type fields of their corresponding responses. Wu et al. [45] use an autoencoder-based semi-supervised feature learning approach for DNS covert channel detection, which can automatically learn the concept of semantic similarity among features of normal traffic, thus avoiding extracting features manually. However, these methods are only applicable to non-encrypted DNS.

TLS Fingerprinting. Transport Layer Security (TLS) protocol is one of the most common security protocols used for encryption. Since packets of the TLS handshake phase contains plain text information, this information is used for TLS fingerprinting. A series of studies [20–22,46] on TLS fingerprinting by Blake Anderson and David McGrew from Cisco in recent years are some of the most representative works. Through the analysis of large-scale TLS flows, they investigate how malware and enterprise applications use TLS, and propose an accurate TLS fingerprinting method using destination context and knowledge bases. Razaghpanah et al. [47] analyze and fingerprint handshake messages to characterize the TLS APIs and libraries that Android apps use and evaluate weaknesses. TLS fingerprints are also used for censorship. Frolov et al. [34] analyze real-world TLS traffic, find that the many popular censorship circumvention tools use TLS configurations that are easily distinguishable from the real-world traffic they attempt to mimic. Chai et al. [48] outline the importance of encrypted server name indication from TLS 1.3 to censorship circumvention.

Encrypted Traffic Classification. Encrypted traffic identification aims to analyze which application the traffic comes from, and has attracted attention from researchers in recent years. Korczyński et al. [49] propose an approach based on Markov chains to model possible sequences of message types observed in single-directional SSL/TLS sessions. Chen et al. [50] propose a multiple-attribute-based encrypted traffic classification system based on the DNS traces generated during the application runtime. Thijs van Ede et al. [51] propose an unsupervised approach for creating mobile app fingerprints from encrypted network traffic.

Privacy of Encrypted DNS. With the encrypted DNS protocol, DNS over TLS (DoT) and DNS over HTTPS (DoH), which have been standardized in recent years, their privacy issues have received extensive attention from researchers. Houser et al. [13] develop a DoT fingerprinting method to analyze DoT traffic and determine if a user has visited websites of interest to adversaries. Siby [14] propose a novel feature set to perform traffic analysis attacks against encrypted DNS and show that traffic analysis enables the identification of domains. Bushart et al. [15] propose a novel traffic analysis method to infer websites a user visits purely based on encrypted and padded DNS traces. Hoang et al. [33] quantify the potential improvement to user privacy that a full deployment of DoH/DoT would achieve in practice.

DoH Tunneling Detection. To the best of our knowledge, [52] is the only work that has discussed DoH tunneling detection as of the writing of this paper. The authors of [52] use the features of DoH flow to construct a neural network classifier to detect DoH tunneling and achieved an accuracy of up to 99% on their dataset. Compared with [52], we have made a lot of improvements in method design, data collection, and experimental setup. In [52], the classical tunneling tool dns2tcp was used and an idealized experiment was performed in the laboratory LAN. In contrast, we used the latest released tool [31,32] to collect more realistic traffic in the Internet environment and find some new insights such as the detection accuracy is related to the geographic location of DNS queries. We use TLS fingerprints for DoH for the first time and do not use some idealized features to build a classifier compared to existing works, which makes the proposed method more practical. Our experiment also discusses the influence of various factors on the detection results in detail, including adversarial considerations by exploring the potential evasion, which has not been discussed in existing works. We have proved through experiments that evading detection is difficult since attackers need to imitate multiple sets of features at the same time, and more importantly, some features are difficult to imitate. Even if the attacker evades the detection, our method can still provide the defender with helpful information for attack investigation.

9. Conclusion

DNS over HTTPS is a great step for privacy, but it is also a giant leap for attackers. The loss of visibility into domain names in DNS lookups has caused obstacles to existing DNS tunneling detection methods. In this paper, we present a novel method to detect DoH-based data exfiltration (DoH tunneling). We analyze TLS fingerprints of DoH clients and build classifiers with flow-based features to detect the DoH tunneling. We find that TLS fingerprints of DoH clients are often unique and can be used for preliminary detection of DoH tunneling. We train and test classifiers with flow-based features. Through a series of experiments, we prove that the proposed method can accurately detect DoH tunneling in most cases. Attackers need to spend a great price, or only in theory, can evade detection. The proposed method can still provide defenders with helpful information for attack investigation even when attackers evade detection. As DoH is developing and being widely deployed, we hope that our findings can help further research to explore security defense mechanisms related to DoH.

CRediT authorship contribution statement

Mengqi Zhan: Conceptualization, Methodology, Software, Writing – original draft. **Yang Li:** Investigation, Writing – review & editing, Supervision. **Guangxi Yu:** Investigation, Writing – reviewing & editing. **Bo Li:** Supervision. **Weiping Wang:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank the anonymous reviewers for their insightful suggestions. This work is supported by the Project of National Key R&D Program of China under Grant No. 2019YFB1005200 and No. 2019YFB1005201.

References

[1] Use of DNS tunneling for C&C communications, https://securelist.com/use-of-dns-tunneling-for-cc-communications/78203/.

[2] UDPoS - exfiltrating credit card data via DNS, https://www.forcepoint.com/zh-hans/blog/x-labs/udpos-exfiltrating-credit-card-data-dns.

[3] V. Paxson, M. Christodorescu, M. Javed, J. Rao, R. Sailer, D.L. Schales, M. Stoecklin, K. Thomas, W. Venema, N. Weaver, Practical comprehensive bounds on surreptitious communication over DNS, in: 22nd USENIX Security Symposium, USENIX Security, 2013, pp. 17–32.

[4] J. Liu, S. Li, Y. Zhang, J. Xiao, P. Chang, C. Peng, Detecting DNS tunnel through binary-classification based on behavior features, in: 2017 IEEE Trustcom/BigDataSE/ICESS, IEEE, 2017, pp. 339–346.
[5] C. Liu, L. Dai, W. Cui, T. Lin, A byte-level CNN method to detect DNS tunnels, in: IEEE 38th International Performance Computing and Communications Conference, IPCCC, IEEE, 2019, pp. 1–8.
[6] R. Tang, C. Huang, Y. Zhou, H. Wu, X. Lu, Y. Sun, Q. Li, J. Li, W. Huang, S. Sun, et al., A practical machine learning-based framework to detect DNS covert communication in enterprises, in: International Conference on Security and Privacy in Communication Systems, SecureComm, Springer, 2020.
[7] D. Perdices, J. Ramos, J.L. García-Dorado, I. González, J.E.L. de Vergara, Natural language processing for web browsing analytics: Challenges, lessons learned, and opportunities, Comput. Netw. 198 (2021) 108357.
[8] P. Pearce, B. Jones, F. Li, R. Ensafi, N. Feamster, N. Weaver, V. Paxson, Global measurement of DNS manipulation, in: 26th USENIX Security Symposium, USENIX Security, 2017, pp. 307–323.
[9] B. Liu, C. Lu, H. Duan, Y. Liu, Z. Li, S. Hao, M. Yang, Who is answering my queries: Understanding and characterizing interception of the DNS resolution path, in: 27th USENIX Security Symposium, USENIX Security, 2018, pp. 1113–1128.
[10] I.N. Bermudez, M. Mellia, M.M. Munafo, R. Keralapura, A. Nucci, Dns to the rescue: Discerning content and services in a tangled web, in: Proceedings of the 2012 Internet Measurement Conference, 2012, pp. 413–426.
[11] DNS over HTTPS (DOH), RFC8484, https://tools.ietf.org/html/rfc8484.
[12] D. Vekshin, K. Hynek, T. Cejka, DoH insight: detecting DNS over HTTPS by machine learning, in: Proceedings of the 15th International Conference on Availability, Reliability and Security, 2020, pp. 1–8.
[13] R. Houser, Z. Li, C. Cotton, H. Wang, An investigation on information leakage of DNS over TLS, in: Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies, CoNEXT, 2019, pp. 123–137.
[14] S. Siby, M. Juarez, C. Diaz, N. Vallina-Rodriguez, C. Troncoso, Encrypted DNS=privacy? A traffic analysis perspective, in: Network and Distributed System Security Symposium, NDSS, 2020.
[15] J. Bushart, C. Rossow, Padding ain't enough: Assessing the privacy guarantees of encrypted DNS, in: 10th USENIX Workshop on Free and Open Communications on the Internet, FOCI, 2020.
[16] M. Trevisan, F. Soro, M. Mellia, I. Drago, R. Morla, Does domain name encryption increase users' privacy? ACM SIGCOMM Comput. Commun. Rev. 50 (3) (2020) 16–22.
[17] First-ever malware strain spotted abusing new DoH (DNS over HTTPS) protocol, https://www.zdnet.com/article/first-ever-malware-strain-spotted-abusing-new-doh-dns-over-https-protocol/.
[18] Iranian hacker group becomes first known APT to weaponize DNS-over-HTTPS (DoH), https://www.zdnet.com/article/iranian-hacker-group-becomes-first-known-apt-to-weaponize-dns-over-https-doh/.
[19] J. Ahmed, H.H. Gharakheili, Q. Raza, C. Russell, V. Sivaraman, Monitoring enterprise DNS queries for detecting data exfiltration from internal hosts, IEEE Trans. Netw. Serv. Manag. 17 (1) (2020) 265–279.
[20] B. Anderson, D. McGrew, TLS beyond the browser: Combining end host and network data to understand application behavior, in: Proceedings of the Internet Measurement Conference, IMC, 2019, pp. 379–392.
[21] B. Anderson, D. McGrew, Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, 2017, pp. 1723–1732.
[22] B. Anderson, S. Paul, D. McGrew, Deciphering malware's use of TLS (without decryption), J. Comput. Virol. Hacking Tech. 14 (3) (2018) 195–211.
[23] DNSSEC, https://www.dnssec.net/.
[24] DNScrypt, https://www.dnscrypt.org/.
[25] DNS over TLS (DOH), RFC7858, myehosthttps://tools.ietf.org/html/rfc7858.
[26] Firefox continues push to bring DNS over HTTPS by default for US users, https://blog.mozilla.org/blog/2020/02/25/firefox-continues-push-to-bring-dns-over-https-by-default-for-us-users/.
[27] A safer and more private browsing experience with Secure DNS, https://blog.chromium.org/2020/05/a-safer-and-more-private-browsing-DoH.html.
[28] Release notes for microsoft edge stable channel, https://docs.microsoft.com/en-us/deployedge/microsoft-edge-relnote-stable-channel.
[29] Windows will improve user privacy with DNS over HTTPS, https://techcommunity.microsoft.com/t5/networking-blog/windows-will-improve-user-privacy-with-dns-over-https/ba-p/1014229.
[30] Z. Durumeric, Z. Ma, D. Springall, R. Barnes, N. Sullivan, E. Bursztein, M. Bailey, J.A. Halderman, V. Paxson, The security impact of HTTPS interception, in: Network and Distributed System Security Symposium, NDSS, 2017.
[31] DNS exfiltration over DNS over HTTPS (DoH) with godoh, https://sensepost.com/blog/2018/waiting-for-godoh/.
[32] Data exfiltration over DNS request covert channel, https://github.com/Arno0x/DNSExfiltrator.
[33] N.P. Hoang, A. Akhavan Niaki, N. Borisov, P. Gill, M. Polychronakis, Assessing the privacy benefits of domain name encryption, in: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, AsiaCCS, 2020, pp. 290–304.
[34] S. Frolov, E. Wustrow, The use of TLS in censorship circumvention, in: Network and Distributed System Security Symposium, NDSS, 2019.
[35] Q. Scheitle, O. Hohlfeld, J. Gamba, J. Jelten, T. Zimmermann, S.D. Strowes, N. Vallina-Rodriguez, A long way to the top: Significance, structure, and stability of internet top lists, in: Proceedings of the Internet Measurement Conference 2018, 2018, pp. 478–493.
[36] J. Ahmed, H.H. Gharakheili, Q. Raza, C. Russell, V. Sivaraman, Real-time detection of DNS exfiltration and tunneling from enterprise networks, in: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management, IM, IEEE, 2019, pp. 649–653.
[37] J. Steadman, S. Scott-Hayward, DNSxD: Detecting data exfiltration over DNS, in: 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN, IEEE, 2018, pp. 1–6.
[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[39] Extension Mechanisms for DNS (EDNS(0)), RFC6891, https://tools.ietf.org/html/rfc6891.
[40] The ENDS(0) padding option, RFC7830, https://tools.ietf.org/html/rfc7830.
[41] Padding policies for extension mechanisms for DNS (EDNS(0)), RFC8467, https://tools.ietf.org/html/rfc8467.
[42] Apple adds support for encrypted DNS, https://www.zdnet.com/article/apple-adds-support-for-encrypted-dns-doh-and-dot/.
[43] Native Linux support for DoH (DNS over HTTPS), https://forums.opensuse.org/showthread.php/536030-Native-Linux-support-for-DoH-(DNS-over-HTTPS).
[44] M. Luo, Q. Wang, Y. Yao, X. Wang, P. Yang, Z. Jiang, Towards comprehensive detection of DNS tunnels, in: IEEE Symposium on Computers and Communications, ISCC, IEEE, 2020, pp. 1–7.
[45] K. Wu, Y. Zhang, T. Yin, TDAE: Autoencoder-based automatic feature learning method for the detection of DNS tunnel, in: 2020 IEEE International Conference on Communications, ICC, IEEE, 2020, pp. 1–7.
[46] B. Anderson, D. McGrew, Accurate TLS fingerprinting using destination context and knowledge bases, 2020, arXiv preprint arXiv:2009.01939.
[47] A. Razaghpanah, A.A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, P. Gill, Studying TLS usage in android apps, in: Proceedings of the 13th International Conference on Emerging Networking Experiments and Technologies, CoNEXT, 2017, pp. 350–362.
[48] Z. Chai, A. Ghafari, A. Houmansadr, On the importance of encrypted-SNI (ESNI) to censorship circumvention, in: 9th USENIX Workshop on Free and Open Communications on the Internet, FOCI, 2019.
[49] M. Korczyński, A. Duda, Markov chain fingerprinting to classify encrypted traffic, in: IEEE Conference on Computer Communications, INFOCOM, IEEE, 2014, pp. 781–789.
[50] Y. Chen, T. Zang, Y. Zhang, Y. Zhouz, Y. Wang, Rethinking encrypted traffic classification: A multi-attribute associated fingerprint approach, in: 2019 IEEE 27th International Conference on Network Protocols, ICNP, IEEE, 2019, pp. 1–11.
[51] T. van Ede, R. Bortolameotti, A. Continella, J. Ren, D.J. Dubois, M. Lindorfer, D. Choffnes, M. van Steen, A. Peter, FLOWPRINT: Semi-supervised mobile-app fingerprinting on encrypted network traffic, in: Network and Distributed System Security Symposium, NDSS, Internet Society, 2020.
[52] M. MontazeriShatoori, L. Davidson, G. Kaur, A.H. Lashkari, Detection of DoH tunnels using time-series classification of encrypted traffic, in: The 5th IEEE Cyber Science and Technology Congress, IEEE, 2020, pp. 63–70.

Mengqi Zhan received the B.S. degree in computer science from Beijing Normal University, Beijing, China, in 2018. He is currently a Ph.D. student at Institute of Information Engineering, Chinese Academy of Sciences. His research interests include network security, network system, network security threat detection.

Yang Li received the B.S. degree in signal and information processing from Harbin Institute of Technology, China, in 2004, the Ph.D. degree in Communication Networking from Kyungpook National University, Korea, in 2009. She is currently an associate professor at Institute of Information Engineering, Chinese Academy of Sciences. Her research interests include 5G network security, network and content security, network big data, network security threat detection, and situation awareness.

Guangxi Yu received his Ph.D. degree in computer science from University of the Chinese Academy of Sciences, Beijing, China, in 2020. He is currently an assistant professor at Institute of Information Engineering, Chinese Academy of Sciences. His research interests include DNS security, 5G network security.

Bo Li received his Ph.D. degree in computer science from Chinese Academy of Sciences, Beijing, China, in 2011. He is currently a Professor at the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include cyber security, database, big data.

Weiping Wang received his Ph.D. degree in computer science from Harbin Institute of Technology, China, in 2008. He is currently a Professor at the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include cyber security and big data.