Computer Networks 197 (2021) 108322

Contents lists available at ScienceDirect

Computer Networks

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

A comprehensive survey on DNS tunnel detection

Yue Wang [a], Anmin Zhou [a], Shan Liao [a], Rongfeng Zheng [b], Rong Hu [c], Lei Zhang [a,*]

[a] School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China,
[b] School of Electronics and Information Engineering, Sichuan University, Chengdu 610065, China
[c] School of Electrical Engineering, Sichuan University, Chengdu 610065, China

ARTICLE INFO

Keywords:
Domain Name System (DNS)
DNS tunnel detection
Threshold-based methods
Signature-based methods
Machine learning-based methods
Deep learning-based methods

ABSTRACT

Domain Name System (DNS) tunnels, established between the controlled host and master server disguised as the authoritative domain name server, can be used as a secret data communication channel for malicious activities. Owing to the ready evasion of the DNS traffic to bypass the network security mechanism, DNS tunnelling can cause severe damage. Thus, an in-depth and comprehensive understanding of the various detection technologies is of considerable importance when facing this type of threat. However, most of the existing reviews focus on a single aspect of the DNS tunnel detection technologies, such as methods based on machine learning, traffic, and payload analysis. In addition, few studies have conducted comprehensive investigation that includes a sequentially integrated range of literature in this researchfield, or have analysed the latest literature on DNS tunnels. This paper reviews these detection technologies from a novel perspective of rule-based and model-based methods with descriptions and analyses of the DNS-based tools and their corresponding features. To the best of our knowledge, this is the first study to comprehensively discuss and analyse DNS tunnel detection in a novel and specific classification fashion from various aspects in detail, covering almost all the detection methods developed from 2006 to 2020. Furthermore, a comparative analysis of detection methods and several suggestions for future research directions are presented.

1. Introduction

Controlling the flow in and out of the network edge is critical to guarantee the security of the information system, which can be effectively restrain malicious traffic attacks [1]. However, some attackers would disguise malicious traffic as legitimate traffic via tunnelling, to bypass the security policy at the network edge and carry out malicious activities. Tunnelling is a packet encapsulation technology that can encapsulate the original packet in another packet of different protocols for transmission [2]. Traditional tunnels are mainly based on transmission or network-layer protocols. Recently tunnelling technology has gradually started being implemented via combining the application layer protocols, such as Hypertext Transfer Protocol (HTTP) [3], Domain Name System (DNS) [4] , and Secure Shell (SSH) [5].

In recent years, the invalidation of network security devices for inspecting DNS traffic and the non-existence of security policies has led to a progressive growth in illegal activities using DNS [6]. DNS, a hierarchical distributed database system, is one of the most critical infrastructures on the Internet [7], and a benchmark for almost all network activities. More precisely, the DNS converts the IP address into a domain name that is easier to understand and remember than the original IP address, with more convenience in accessing various resources on the Internet. Conspicuously, the DNS is not intended for data transmission; thus, the configuration of the firewall, that allows the user datagram protocol(UDP) 53 port data utilised by the DNS service is the default [8]. In other words, the DNS traffic can travel across the network edge unimpededly without the need for tight security strategies. Furthermore, the host often unconditionally trusts the response information of the DNS server [9]. In summary, the above aspects make DNS an ideal candidate for tunnelling.

In 1998, the DNS tunnel for data transmission was discovered for the first time [10], and it was initially designed to bypass the network edge to obtain free Internet access [11]. Owing to the efficacy of bypassing the network security mechanism, an increasing number of services tend to use the DNS tunnel. For example, in 2007, Trend-Micronic proposed a method to distribute updated malicious code signatures through DNS tunnels [12], the purpose of which is to provide a signature update service to the anti-virus client. Nevertheless, few cases of exploiting DNS tunnels are legal and benign. According to the corresponding investigations, nearly half of the enterprise networks are assessed as existing DNS tunnelling with high probability [13]. As early as 2012 at the RSA conference, E. Skoudis pointed out that the DNS-based remote control malware would be one of the six most

* Corresponding author.
E-mail address: zhanglei2018@scu.edu.cn (L. Zhang).

https://doi.org/10.1016/j.comnet.2021.108322
Received 4 January 2021; Received in revised form 7 July 2021; Accepted 11 July 2021
Available online 17 July 2021

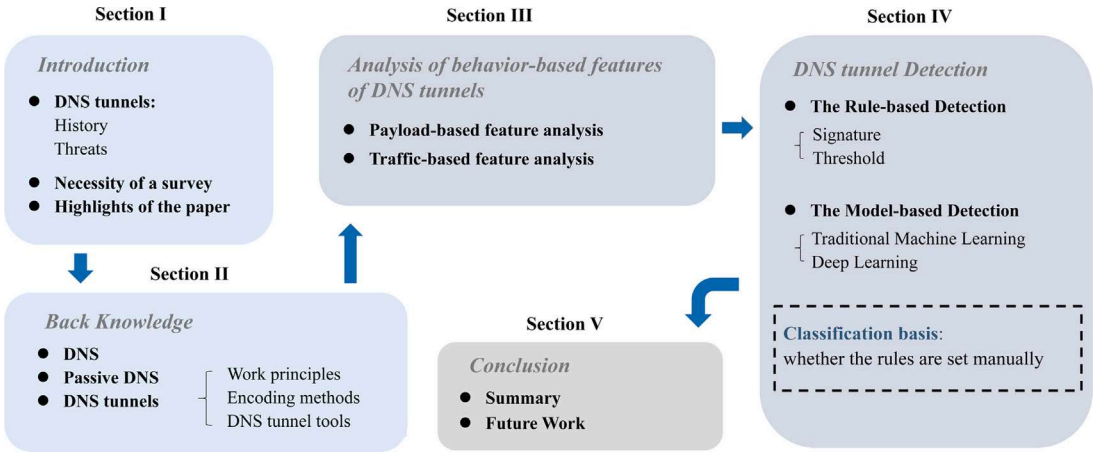

Fig. 1. The structure of the paper.

dangerous network attacks in the future [14]. In addition, the DNS tunnel can be used for remote command and control information transmission of malware, data leakage, and so on [11]. Even though DNS tunnels only provide low bandwidth for data transmission, the attackers still can steal data or maintain communication with malware through the tunnel [15]. Fedderbot [16], Morto [17], and Wekby pisloder [18] highlighted the fact that there is numerous malware utilising the DNS tunnel to send command and control (C2) instructions and transmit attack payloads. For example, the American retailer Home Depot has previously been attacked by a piece of malware named FrameworkPos [19], which is used to capture payment card data via DNS requests. More than 56 million credit card accounts and customer debit information were stolen. Therefore, it is imperative and inevitable to deal with the DNS tunnel detection technology in both the academic and industrial domains.

Hitherto, some progress concerning DNS tunnel detection has been made. However, few studies have discussed this type of technology from a comprehensive perspective. Both [20] and [21] focus on machine learning-based DNS tunnel detection, mainly on the comparative analysis of various machine learning algorithms' performance in this scenario. Compared with the previous two papers, [22] is more broad and divides detection technology into three categories: flow analysis, payload analysis, and mixed analysis, based on data types. Unfortunately, several drawbacks exist in the above studies. The unspecific classification and deficient references yield a defective overview of the current DNS tunnel detection. Similarly, [23] with the categories of flow detection and payload detection is defective because of the limited elaboration of machine learning-based technologies. In this study, the current DNS tunnel detection technology will be investigated and analysed from a more comprehensive perspective to help researchers obtain relevant knowledge, scientifically and systematically, and to provide inspiration for future research. For a better understanding, the overall framework of this study is presented in Fig. 1.

The remainder of this paper is organised as follows. Section 2 provided background information on the DNS tunnel detection, including the DNS protocol, passive DNS, and DNS tunnel technology. Then, the common characteristics of DNS tunnel detection are analysed based on three different perspectives, that is, payload, flow, and session, as described in Section 3. Next, detailed discussions concerning the different types of DNS tunnel detection are presented in Section 4. Finally, Section 5 summarises this paper and expounds on future research. Moreover, compared with other DNS tunnel detection technology reviews, the highlights of this study are as follows.

(1) To the best of our knowledge, this study is the first comprehensive review of DNS tunnel detection in all aspects. Unlike previous research, payload and traffic analysis are only applied to describe features in DNS tunnel detection. A variety of detection methods are discussed in a comprehensive and novel approach, covering almost all detection methods developed from 2006 to 2020.
(2) A novel classification of DNS tunnel detection is proposed in detail, which is a rule-based and model-based method on the condition of the manual setting rule. The rule-based method is subdivided into signature and threshold methods, whereas traditional machine learning and deep learning constitute the model-based method. From this point, a sequentially integrated range of literature in this research domain is reviewed comprehensively, including the most recently published.
(3) The advantages and disadvantages of these different detection methods and their application environments are analysed in detail, including several future research directions.

2. Background information

This section explains the relevant background information on DNS tunnel detection to enable a deeper understanding of this domain. First, the basic knowledge on DNS is introduced, including the working principles of the DNS protocol, message formats, and the concepts of passive DNS technology. DNS tunnel-related knowledge is described next. In addition to, the introduction of the DNS tunnelling technical route and classification, the comparative analyses of DNS tunnel tools are also summarised. Different DNS tunnel tools would generate slightly discrepant tunnels owing to the distinctions in the technical implementation. These distinctions also have a certain influence on DNS tunnel detection and are worth summarising and analysing.

2.1. DNS

DNS is a distributed database system that maps domain names and IP addresses to each other. Routers identify hosts by their IP address, while people tend to use hostnames that are easier to remember to identify hosts [24]. A domain name containing multiple hostnames or even subdomains is the unique name of a server or network system on the Internet. It is a sequence of labels that present one level's domain name, consisting of letters, numbers, and connectors, and separated by periods. Both the domain name levels and domain name servers are hierarchical tree structures. The top of the tree indicates that the root domain name is empty, followed by the top-level name, second-level name, and so on. These domains are stored separately in the distributed domain name servers. Furthermore, the corresponding servers are root, top-level, and authoritative name servers. The local name server that does not belong to the domain name server hierarchy can cache the records to avoid querying other servers.

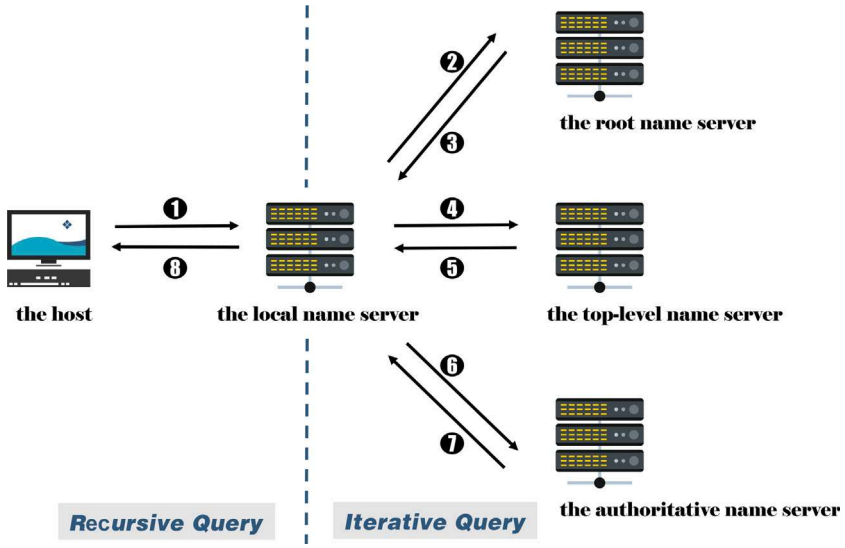

Fig. 2. The procedure of domain name query.

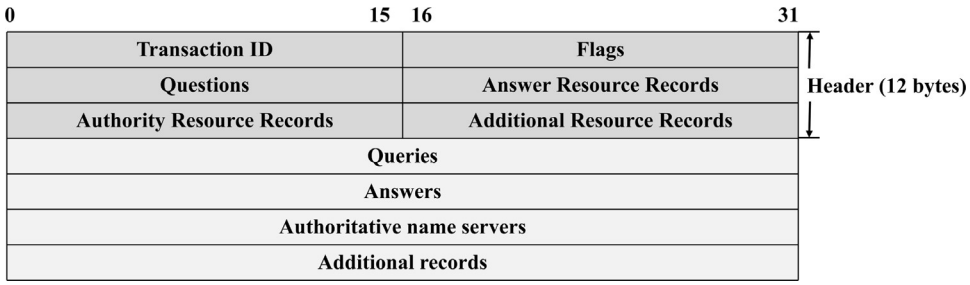

Fig. 3. The DNS messages format [25].

There are two query methods for domain resolution: iterative and recursive [24]. Iterative query refers to the local name server proxy host to communicate directly with an authoritative name server, top-level name server, and root name server successively [26]. Nevertheless, a recursive query starts at the local name server and queries a more advanced domain name server one by one to obtain the last IP address. The current domain name query process is shown in Fig. 2, which usually combines these two methods. Between the host and the local name, the server is a recursive query, whereas between the local name server and other servers is an iterative query. The query process is as follows: the host first asks the local name server and returns the response immediately if it has the cache. Otherwise, the local name server would proxy the host to obtain the IP address according to the recursive query.

During the query process, described above, only two types of DNS packets would emerge, namely DNS query and response packets with the same format, as shown in Fig. 3. The first 12 bytes of the message are the header area, followed by the question, answer, authority, and attachment areas. The DNS query uses the question area to pass the query information, whereas the answer area's resource record field is for the DNS response to reply. Each DNS response contains one or more resource records that map hostnames to IP addresses. The Type field value in the resource record determines the type of DNS query, parts of which are shown in Table 1.

2.2. Passive DNS

Passive DNS, proposed by Florian Weimer in 2004, is the reverse access to query DNS corresponding information [30]. It replicates DNS activities and reconstructs the DNS data, available in the global domain name system, into a central database for researchers to retrieve and query. These obtained data embody not only the current query information, but also the historical records. The introduction of passive DNS is of great value to researchers in identifying malicious domain names, detecting phishing sites or tunnels, and other related domains.

The principle of passive DNS technology, shown in Fig. 4, is based on passive monitoring of DNS queries and response packets. The passive sensor is usually installed on the recursive DNS server to capture the original packet from the network interface. Ordinary users obtain the relevant resource record of the specific domain name through a DNS query, and the passive DNS sensor duplicates these data into a dedicated database simultaneously, hereinafter referred to as DNSDB. Hence, this makes it easy to query data that could be extremely difficult or even impossible to obtain solely through DNS protocol, such as all the subdomains under the main domain name [31], DNS history query of the domain name, and so on.

2.3. DNS tunnels

First, it is important to clarify the definition of DNS tunnels. The DNS tunnel is a type of tunnel technique, based on the DNS protocol, which utilises the DNS query process and encapsulates the data in the DNS query/response package to build a proprietary tunnel for transmission communication between the sender and receiver. DNS tunnelling is also known as DNS covert channels, DNS exfiltration, and even DNS steganography. DNS tunnelling and DNS covert channels have the highest usage frequency, followed by DNS exfiltration, and the usage of DNS steganography is the lowest. From the aspect of technical principles, DNS tunnelling and DNS covert channels are more appropriate names than others, of which DNS tunnels are superior.

Table 1
Several examples of Resource Record (RR) types.

Resource Record (RR) types	Value	Meaning	RFC
Address (A)	1	The IPv4 address	
Name Server(NS)	2	The authoritative name server	
Canonical Name (CNAME)	5	The canonical name for an alias	
Start of Authority (SOA)	6	Marks the start of a zone of authority	1035[25]
NULL	10	The null RR (experimental)	
Pointer (PTR)	12	The domain name pointer	
Mail Exchange (MX)	15	Mail exchange	
Text (TXT)	16	Storing text data on a domain	
AAAA	28	The IPv6 address	3596[27]
KEY	25	Storing a public key associated with a domain	2535[28]
Service (SRV)	33	Service locator	2782[29]

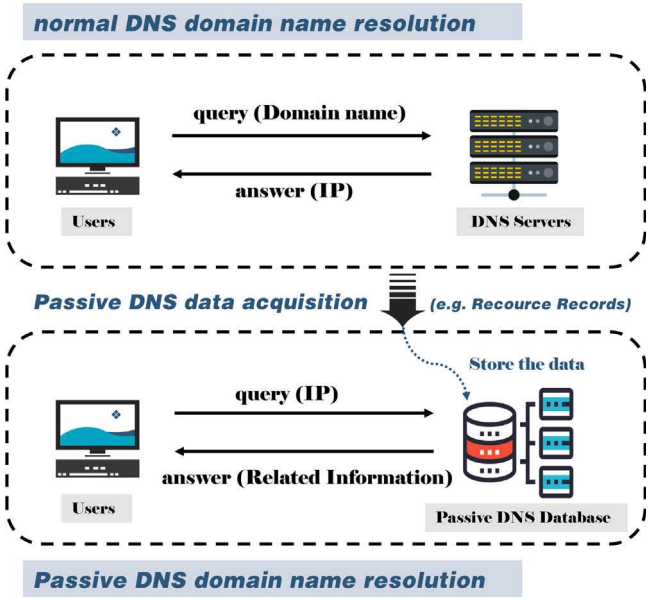

Fig. 4. The technical framework of passive DNS.

Essentially, DNS tunnelling is a bidirectional communication process, during which the client sends data to the server (the process of data exfiltration) and vice versa. Therefore, the name of the DNS exfiltration is not sufficiently accurate. Technically, DNS steganography does not only refer to the sending of confidential information in the tunnel. Thus, it is rarely used in this field, and is somewhat inappropriate. For instance, the transmission of confidential information through the time–frequency of DNS packets is also a type of DNS steganography, but outside the scope of our research. Hence, to improve the readability of this study, the umbrella term "DNS tunnelling" is used.

Second, the type of DNS tunnelling varies depending on the method of data encapsulation. For example, a DNS tunnel encapsulates data in the relaxation space, which is an unused but unreasonable space in a packet. Nonetheless, it does not receive much attention from experts because of its vulnerability to detection. To the best of our knowledge, most researchers have focused on DNS tunnelling using domain name fields to encapsulate data. Therefore, the analysis and detection technologies discussed in this paper are similar to this type of DNS tunnel. Meanwhile, to ensure comprehensiveness, we regard the DNS tunnel using relaxation space as the avoiding technology of mainstream DNS tunnelling detection. The working principle and detection technologies are discussed and summarised in Section 4.3.

As mentioned in the introduction, DNS tunnelling is not always malicious and can be used for legitimate purposes, such as transmitting small amounts of necessary business data [12]. However, in most cases, a DNS tunnel is used for malignant activity. As legitimate DNS tunnelling always performs legal and known data transmission on the premise that both the client and server reach a consensus, there is no need for detection. In addition, suppose that such a legitimate DNS tunnel exists within an organisation's network. In that case, a simple whitelist technology could reduce the negative impact of the detection of other suspected unknown malicious DNS tunnelling.

This section explains the relevant background information on DNS tunnelling from the aspect of the DNS tunnel technical route, encoding methods, and implementation tools.

2.3.1. *Working principles*

DNS tunnel implementation requires three parts: the domain or subdomain controlled by the attacker, DNS tunnel tools, and camouflaged DNS server. The C2 server, disguised as an authoritative DNS server by the attacker, can be accessed through this domain. Meanwhile, the DNS tunnel tool is generally in C/S mode [32], where the client is installed on the controlled hosts , and the server is on the camouflaged DNS server [33]. Depending on the communication method between the controlled host and camouflaged DNS server, the DNS tunnel can be divided into direct connection mode and relay mode.

DNS tunnelling in direct connection mode refers to the fact that the controlled host connects directly to the specified target DNS server. The data encapsulated in DNS packets, in a certain encoding manner, can be directly sent to a specific server instead of creating an iterative or recursive query. Therefore, a higher communication efficiency can potentially be achieved with the disadvantage of effortless discovery owing to its low concealment [34]. In addition, the institutional network usually forbids the host to specify the DNS server by itself. In this case, once direct connection communication occurs, it most likely implies the existence of a DNS tunnel in the direct connection mode.

Compared with the former, the communication principle of relay-mode DNS tunnelling is more complex. The basic idea is to use a disguised DNS server as a relay node and then establish a tunnel using the normal DNS query process for data transmission [35]. The technical framework of the DNS tunnel is shown in Fig. 5. Assume that the disguised DNS server's IP address in the external network is A.B.C.D, and the domain name, registered in .com top-level domain name server, is "tunnel.com". Before data transmission, both sides of the communication need to activate the DNS tunnel tool first, and the two sides would negotiate mechanisms such as data coding during this process. The detailed data communication process is as follows.

(1) The controlled host constructs the uploaded data into the subdomain name requested in the DNS query, such as "updata.tunnel.com', and then sends it to the local name server.
(2) Because of the cache, the local name server would generally ask the top-level name server directly to obtain the IP address of the authoritative name server named tunnel.com , which is disguised by the C2 server.

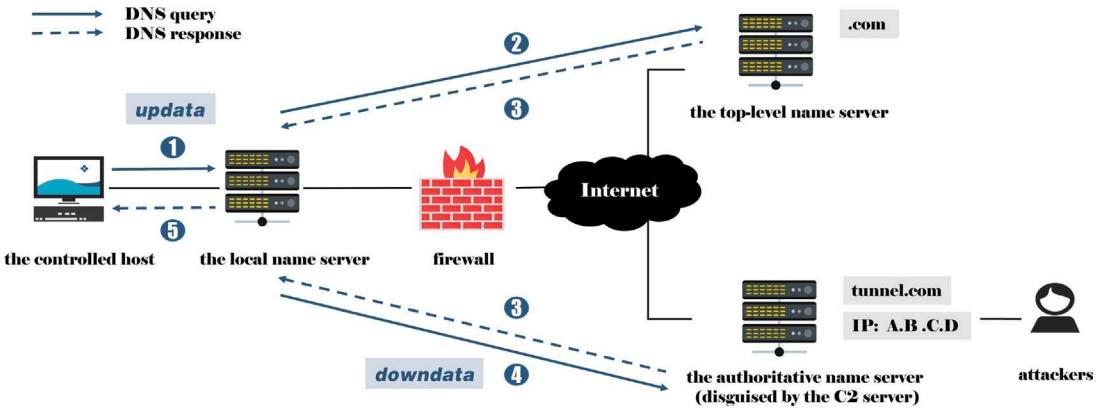

Fig. 5. The technical framework of DNS tunnels.

(3) The .com name server sends the IP(A.B.C.D) of the disguised authoritative name server to the local DNS server.
(4) The local DNS server sends a query of updata.tunnel.com to the IP address. Then, the C2 master can extract the updata from the carefully constructed packets.
(5) The C2 master sends several command control instructions as downdata back to the local name server by wrapping them in DNS response packets.
(6) Finally, the local name server transmits the DNS response to the controlled host. Thus the two-way communication between the controlled host and the C2 master is accomplished.

Despite the differences in work principles between these two types of DNS tunnels, they are implemented based on the DNS protocol to build a tunnel for data transmission. Overall, although the direct mode DNS tunnel has a higher communication efficiency, it can be easily defended by security policies such as IP blacklist or whitelist [34]. By comparison, the relay-mode DNS tunnel, which is popular among the attackers and researchers, has more roubust survivability and higher concealment. Given the above considerations, the main objective of this study is the relay mode DNS tunnel, which is directly referred to as a DNS tunnel in this paper.

2.3.2. Encoding methods

DNS tunnelling uses a DNS query and response packets to transmit the data. The subdomain name queried in DNS packets can be used to encapsulate upstream data. According to RFC1034 [36], the maximum length of a domain name is 253 bytes, including the separation symbol. Each label's maximum length is 63 bytes, separated by a dot. The characters used in a domain name could be letters (case insensitive), numbers, or connectors, meaning that each character could have 37 different values [33]. However, the DNS response is used for downstream data transmission, mainly by utilising the data portion of the resource record field. Before data transmission, encoding the data to meet the standard requirements is needed.

Base32 and Base64 are two of the most commonly used data encoding technologies. Base32[37] encodes arbitrary byte data using 32 printable characters, including case-intensive letters a-z and numbers 2-7. Each byte encodes five bits of raw data applicable to 37 available characters per character in the domain name. Unlike the scenario where Base32 is mainly used for uploading data, Base64[37] is typically used for downloading data. It is based on 64 printable characters, including case-sensitive letters a-z, A-Z, numbers 0–9, and the remaining two inconsistencies between different systems. When adopting this method each byte encode six bits of raw data. This encoding method is usually used for the Text (TXT) record of the DNS response, in that the TXT record allows any text in response which also requires letters to be case-sensitive [38]. The truth is that some record types , such as TXT, are uncommon in the normal DNS traffic. Table 2 [39] shows a record type distribution, under normal behaviour, obtained from an Internet service provider (ISP) generated dataset. It can be seen that TXT, anonical name (CNAME), service (SRV), mail exchange (MX) account for a small proportion. This means that they are uncommonly used record types. It is noteworthy that DNS tunnel tools tend to use less common record types because more bandwidth is provided compared to others [40]. Furthermore, the types of records used among the DNS tunnel tools are not the same.

Table 2
The distribution of several RR types [39].

RR types	Proportion
%A	0.49
%NS	0.003
%CNAME	0
%SOA	0.0011
%NULL	0.0001
%PTR	0.2728
%MX	0.093
%TXT	0.0426
%AAAA	0.085

Apart from Base32 and Base64, binary encoding is not limited to a finite character set. It could significantly improve data transmission efficiency [41], under the condition where both sides of the communication support it [34]. Moreover, the encoding methods used for DNS tunnel tools are usually different from each other. The encoding technologies mentioned above are the most common, and other data coding methods are not detailed within this article.

2.3.3. DNS tunnel tools

DNS tunnel tools are generally in client/server mode. The client was installed in the controlled host of the target network. Simultaneously, the server is set on the C2 server of the external network and is generally disguised as the authoritative domain name server of the controlled domain [42]. Despite the same core principles, the implementation details of various DNS tunnel tools are differ slightly. According to the abstraction layer, that encapsulates data, DNS tunnel tools can be divided into IP over DNS tunnels and TCP over DNS tunnels [43].

IP over a DNS tunnel encapsulates IP packets in the DNS tunnel. NSTX [45] was one of the first tools to implement IP over DNS, designed by Florian Heinz and Julien Oster in 2000. It requires the client and server to have special kernel configurations, and it can only run on Linux and use Base64 to encode communication data with mostly TXT record types. In addition, Dnscat2, Iodine, and TUNs are also IP over DNS tunnel tools. Dnscat2 and Iodine support both the direct and relay communication modes. In contrast to other tools, Dnscat2[46] is primarily designed to create an encrypted C2 tunnel based on DNS,

Table 3
Summaries of DNS tunnel tools.

Classification	DNS tunnel tool	Encoding methods	Query types	Extension mechanisms for DNS (EDNS(0)) [44]
IP over DNS	NSTX	Base64	TXT	×
	Dnscat2	Hexadecimal	A, AAAA, CNAME, MX, TXT	×
	Iodine	Base32, Base64, Base128	A, CNAME, NULL, MX, TXT, SRV	√
	TUNs	Base32, Base64	CNAME	×
TCP over DNS	Dns2tcp	Base64	TXT, KEY	×
	OzymanDns	Requests:Base32 Responses:Base64	A, TXT	×
	Heyoka	Binary	TXT	√

which uses the hex encoding method. Iodine [47] is the most well-known tool that covers a wide range of platforms and supports a variety of record types and encoding. Iodine also uses extension mechanisms for DNS (EDNS(0)) [44] which allows DNS packets to exceed 512 bytes long [43]. Additionally, both Iodine and NSTX use a mechanism similar to IP fragmentation, which splits and encapsulates IP packets into DNS packets separately, then recombines them at the final destination. However, TUNs [48] do not split IP packets and only use the CNAME record.

TCP over a DNS tunnel encapsulates TCP packets in the DNS tunnel. An earlier example was OzymanDns [49], written by Dan Kaminsky in 2004, which uses Base32 encoding for DNS queries and Base64 encoding for DNS responses. However, this tool lacks stability and is easy to crash. Heyoka [50] used binary encoding as well as EDNS(0) [44]. Relatively speaking, Dns2tcp [51] is more popular, which supports KEY and TXT record types and uses Base64 encoding.

The implementation details of the above tools, such as record types and data coding, are slightly diverse, as shown in Table 3. The √ and × in EDNS(0) column indicate the support and non-support of EDNS(0), respectively. These differences may have an impact on specific detection technologies. For example, TUNs only use CNAME records so that TXT type corresponding features would not work in detection. Therefore, more characteristics of these tools need to be understood to increase the versatility of the detection technology. It is worth noting that not only could these tools generate DNS tunnels , but also some malware could do the same thing.

3. Analysis of behaviour-based feature of DNS tunnels

The typical features of DNS tunnel detection can be divided into two categories: payload analysis and flow analysis. Payload analysis extracts features from a more specific microscopic perspective by analysing the payload information of a single packet or several packets. The traffic analysis assesses overall DNS traffic information in a period, from a more global perspective, to extract effective features. DNS tunnel detection technology can also be classified into two categories based on the features of the method. However this is only a relatively rough classification, without a good overview of the characteristics of the relevant detection technology.

3.1. Payload-based feature analysis

Payload analysis is for a single or several DNS packets, and its focus is on analysing the relevant characteristics of the payload. For example, the signature of specific DNS tunnel tools, request domain name-related features, and so on. Such features are also of interest to the current detection technology. In this section, some commonly used payload features are analysed and described.

- Many valuable features can be obtained through statistical analysis, such as the size of the packet, payload upload/download ratio, etc. As follow:

  (1) **Size of packets**: The upstream data of the tunnel encapsulated in the request packet's domain name field are usually used to leak a large amount of private data. Hence, the size of packets in DNS tunnels is larger owing to efficient data transmission. Conversely, the size of the response package is available and untypical. The downstream data encapsulated in response packets are usually used for transmitting command and control instructions so that the size of these packets would be smaller than or equal to the normal packets. A normal DNS response packet often carries multiple resource records; therefore, the packet size is larger [35]. Overall, the size of the request packet may be a more important feature than that of the response packet. A simple real-time detection system can set an alarm threshold according to this feature. In addition, this type of feature can also be avoided. By reducing the amount of data carried in a request packet, as well as increasing the number of malicious request packets, the attacker can disguise malicious traffic being legitimate and transmit the same amount of data as before. Such circumvention would reduce the availability of the feature and lead to a low transmission performance of the DNS tunnelling. Moreover, in this circumstance, the soaring number of requests in the same domain would become a vital indicator for detection. Hence, the circumvention of this feature is challenging for an attacker who wants to obtain a DNS tunnel with ideal transmission performance, which makes it a typical and decisive feature.

  (2) **The ratio of upload and download payload**: The size of the uploaded data is often more extensive than that of the downloaded data, as shown in the previous description. In that case, there may be a DNS tunnel when the value of this feature is high. This feature is derived from the size of the packets, which combines the size of the request and response packets in DNS tunnelling. Indeed, it has the same limitations as to the size of the packets.

- The domain name receives the most attention in payload analysis. It is usually a text with high readability to enable memorisation. In contrast, the data would be encoded into as many subdomains as possible, so the DNS tunnel's domain name is poorly readable and complex. Because DNS tunnelling works by using domain names for data transmission, differences between the tunnel domain names and normal domain names are inevitable. Generally, the features are extracted from the length of domain names, domain name labels, character entropy, etc. In theory, these features can be circumvented by improving the readability of transmitted data after encoding and disguising themselves as normal domain names. Nonetheless, such an effective circumventing technique has not yet emerged in the current investigations. Therefore, this type of feature based on domain names is of high availability and importance, and related investigations are gaining momentum. Many researchers have explored effective features, from various aspects, by analysing the differences between the tunnel domain names and normal domain names. As follow:

(1) **Length of domain/subdomain name**: The controlled domain name does not change, so it is feasible to detect the length of the subdomain name.
(2) **Number of subdomains/labels**: Each label in the domain name has a maximum length of 63 bytes. When large amounts of data are entered into the domain name field, the number of subdomains or labels increases. It is likely to have multiple, maximum 63-character long labels in the domain name of DNS tunnelling [52].
(3) **The ratio/number of special characters in the domain name**: In the encoded data of DNS tunnelling, the proportion of special characters such as numeric characters and connectors would be higher, as would the proportion of uppercase letters [8].
(4) **The ratio of the longest meaningful substring/word's length in the domain name**: [35] uses the feature of the longest meaningful word over the domain length average. The first is to find the longest meaningful word in the domain name. The length of this word is then divided by the corresponding subdomain to obtain the average value. This feature can improve the judgement of the domain names' readability. Besides the longest meaningful word, the longest meaningful substring is also practicable [53].
(5) **Entropy of characters**: The specificity of encoded data makes valid information available by character frequency analysis. Normal domain names are of high readability and easy to remember, following Zipf's law [54], which states that the frequency of any word is inversely proportional to its rank in the frequency table. In other words, the character frequencies of a normal domain name are concentrated in a few high-frequency characters. Therefore, the random distribution of tunnel domain names makes its character entropy higher than that of the normal domain name. The entropy of the subdomain or hostname is more valuable because of the normality of the top-level and controlled domain names.

- Other valid features are as follow:

(1) **Uncommon record types**: As mentioned in the previous section, record types have different proportions distributions in normal DNS traffic. Table 2 shows the record types, such as record A (49%), record PTR (27.28%), record AAAA (8.5%), and record MX (9.3%) were familiar, while record TXT (4.26%), record SRV (0.01%), record CNAME (0%), and other infrequently used records accounted for less than 10% of the total. Nevertheless, DNS tunnel tools tend to use uncommon record types, considering that these generally provide more bandwidth for transmitting data [40]. Because most DNS tunnel tools support TXT records, the features of TXT records are widely used in most detection technologies. However, it could lose its effectiveness when a record type such as TXT is specified, and the DNS tunnel is established by TUNs tools which only use the CNAME record type. Therefore, it is more appropriate to set a subset of uncommon record types rather than using a specific one. If the record type of the packet belongs to this subset, then it is suspected.
(2) **Signatures of specific DNS tunnel tools**: Part of DNS tunnel tools, such as NSTX, would leave the specific signature in the header or payload of packets. Using this feature, it is possible to identify the existence of DNS tunnels constructed using a specific tool. This limitation is time-consuming and labour-intensive according to the preliminary research on signatures. In addition, it could not detect unknown DNS tunnelling, that is, DNS tunnelling out of the signature library. This feature is the backbone of signature-based detection methods; therefore, more relevant information is described in Section 4.1.1.
(3) **Policy violation**: This feature is used for direct-connected DNS tunnel detection. With the security policy that the intranet is limited to communicating with the trusted DNS server set, the existence of a direct connection between the host and unfamiliar DNS server could suggest a directly connected DNS tunnel. This feature is specialised for certain situations; therefore, it has great limitations. For instance, it is available only if the relevant policies are set in the system network and only for detecting suspicious DNS tunnels in direct connection mode.

3.2. Traffic-based feature analysis

Traffic analysis focuses on the inquiry of the overall DNS traffic, based on the time scale, concentrating more on the abnormal variation of traffic. It stands at a more macroscopic angle to analyse the features of the DNS tunnel. Therefore, in addition to traffic-related features, some additional useful information is included (referred to as traffic features here). Most traffic-based features rely on a certain time window; therefore, they are generally utilised for non-real-time detection. For real-time detection, payload-based features are used more frequently. In this section, these common traffic features are described in detail as follows.

(1) **Volume of DNS traffic to the IP address**: Because the size of the DNS packet is limited to 512 bytes; the tunnel needs multitudinous packets to meet the transmission demand. Consequently, the DNS traffic of the controlled host would be larger than that of the other normal hosts. Consequently, the abnormally high DNS traffic at a certain IP address might indicate the existence of a DNS tunnel. Nevertheless, this feature's weakness lies in the fact that the attacker can utilise the source address IP spoofing to eschew the detection. This means that the attacker can reduce the abnormal DNS traffic of a certain IP by distributing the DNS traffic into different IPs. However the average DNS traffic in this system network may be unusual.
(2) **Volume of DNS traffic to the domain**: Similar to the last feature, numerous DNS traffic would go to the specific domain during data transmission of the tunnel. Similarly, the attacker might employ multiple domain names to equalise traffic and avoid detection, which would degrade the significance of the feature in detection progress.
(3) **Volume of hostnames to the domain name**: Owing to the uniqueness of subdomains for data transmission, the volume of hostnames in a certain tunnel domain could be large, and query volumes for these hostnames could be extremely low. Accordingly, the combination of these features could result in better identification performance. As DNS tunnelling depends on subdomains to transmit data, the momentousness of the volume of hostnames to the domain name is axiomatic. Furthermore, these different hostnames could be further studied from the perspective of payload-based features, which makes this feature more relevant to other features.
(4) **Time interval**: This feature refers to the time between querying a domain and receiving the response. In practice, domain name resolution uses cached records from the local DNS server [40]. It requires a shorter time than the DNS tunnel, which needs to transmit data to the disguised server using an iterative query method. Long time intervals can also occur under normal circumstances when the local DNS server has no cache. However, owing to the uncommonness of this situation, the long average response time interval in the same domain may indicate the existence of DNS tunnelling.

(5) **Domain history**: This feature is often applied in the detection technology of malicious domain names. Through domain history such as historical resolution records, it is possible to determine whether the domain name has participated in malicious activities [17], which is undoubtedly applicable to DNS tunnel detection. Information such as the time added to NS records is significant, as the DNS tunnel typically does not register relevant information very early in advance. This feature is not the fundamental pillar of DNS tunnel detection, which is irrelevant to the working principle of the DNS tunnelling itself. It measures the possibility of DNS tunnelling based on the suspicious nature of the domain name used. Unquestionably, an attacker could choose the domain names that had not been previously involved in any malicious activities.

(6) **Geographic location of DNS server**: The DNS server here refers to the authoritative domain name server that provides the final IP address. Broadly speaking, the phenomenon that a company with no transnational business finds the internal DNS traffic worldwide may suggest the existence of DNS tunnels [55]. When the normal requests in an organisation are transnational, or if the attacker pays special attention to the geographical location when setting up the authoritative domain name server, the applicability of this feature would decline considerably.

(7) **DNS queries to the dynamic domain name server (DDNS)**: DDNS is a service that maps a dynamic IP address to a fixed domain name resolution, such as dyndns, xname, and noip. Once a user connects to the network, the client transfers the host's dynamic IP address to the server, which is responsible for providing DNS service and implementing dynamic domain name resolution. It is convenient for DNS tunnel tools to use DDNS to map the domain name to the IP address of the C2 master [56], and so they do.

(8) **Isolated DNS queries**: Other actions in actual traffic usually accompany DNS requests. For example, when requesting a Web service, HTTP requested to the IP address is followed by a DNS query and response with an IP address [55]. Obviously, most of the tunnel traffic is isolated DNS query packets, in that there would be no request for other network services. Some legitimate isolated DNS queries used by security devices and IP address finder [55] are exceptional cases. In the meantime, the attacker could eschew it by monitoring other activities such as the request for a Web service and then sending a DNS request, which is indeed a shortcoming of this feature.

(9) **Volume of NxDomain response**: The answer to DNS response in DNS tunnelling is less common. Some DNS tunnel tools would respond to domain name queries with NxDomain packets [22], which means that the domain name is non-existent. The existence of a few NxDomain packets is expected but an abnormality occurs with a large increase of them.

(10) **DNS traffic visualisation**: In effect, DNS traffic visualisation is a technology that assists detection rather than an applicable feature. Through the visualisation of relevant features, the intuitive pattern of data variability could be presented based on a time scale, facilitating research and discovery [57]. More information is provided in Section 4.

4. DNS tunnel detection

The basic approach of DNS tunnel detection is to combine feature filtration with manual analysis, as presented in [58,59]. DNS traffic is collected by passive DNS and then filtered using certain features after the pre-treatment process. Finally, the suspicious DNS tunnel was determined by professionals using manual analysis. This type of detection, requiring numerous iterations and the skills of professionals, is not suitable for a high-throughput network, and its detection accuracy depends more on experienced professionals.

Instead of detecting with great uncertainty, the approach of designing rules to detect relying on models or certain platforms is more scientific and efficient. According to whether the rules are set manually, DNS tunnel detection is divided into two categories, namely rule-based and model-based detection, as shown in Fig. 6.

4.1. The rule-based detection

Rule-based detection refers to manually set rules with the analysis of relevant features. Then, the existence of DNS tunnels is determined once the matching of preset rules in the monitored traffic occurs. Its emphasis lies in the rule design of certain features, which can be subdivided into two categories: signature-based and threshold-based methods. This detection technology generally combines various platforms to analyse and detect traffic, which can fall into four categories: network traffic analysis software (e.g. Zeek [60]), big data search components (e.g. Splunk [61], ElasticSearch [62]), firewalls (e.g. PaloAlto Network [63]), and intrusion detection systems/intrusion prevention systems (IDS/IPS) (e.g. Snort [64], OSSEC [65], Sguil [66]).

4.1.1. The signature-based methods

The signature-based method detects a DNS tunnel by matching specific signatures. Its priority is the effective signatures analysed by professionals: static and fixed data features of malicious packets. DNS tunnels can be accurately detected with low false positives via these signatures. Deep packet inspection is utilised to analyse the available signatures of DNS tunnel traffic, which is normally a specific attribute of the DNS header or data content in the payload. Moreover, different DNS tunnel tools with various malicious behaviours can lead to varying degrees of difficulty in analysing sundry signatures.

In 2006, the early DNS tunnel tool NSTX was found to have a unique hard-coded value in DNS packet headers designed as an NSTX signature, based on the Snort IDS by Van Horenbeeck [67]. Although NSTX is being gradually phased out due to its complex configuration and flexibility, Iodine, Dns2tcp, and DnsCat2 are more active at present.

There is plentiful research on Iodine, such as [68,69] that mention the signature of the Iodine and associated specifically conducted experimental analysis. [68] focused on the distinctions of the average packet size between the active and passive DNS tunnel traffic and identified the DNS tunnel mixed with legitimate traffic using open-source Iodine signature rules. [69] studied three usage scenarios of Iodine, corresponding to a mechanism similar to IP fragmentation, a twin tunnel with an SSH connection, and a NULL record type of DNS tunnel traffic. Fortunately, a signature is found in a 13-byte offset of the NULL type packet, which is the same as the initiation of the IP packet (0×4510). Combined with the signature mentioned in [68], the system is successful in the detection of DNS tunnels.

Furthermore, [70] focuses on the SSH signature encapsulated in tunnel traffic. They decoded the base64 twin tunnel data of DNS and SSH generated by Dns2tcp in Wireshark, and found string "AAACCFNTSC0yLjAtT 3BlblNTSF83LjJwMiBVYnVudHUtNHVidW50d TIuMg0" means "SSH - 2.0 - OpenSSH_7. 2 p2 Ubuntu - 4 ubuntu2. 2." that clearly indicates the establishment of an SSH connection [70]. Using this signature, the DNS tunnel with the operation of the SSH connection can be detected. To enhance this signature's universality, the string "T3BlblNTSF8" equivalent to "OpenSSH_" is also available. However, its false negative is high (10%) because of the absence of multitudinous SSH signature variants. The signature analysis is mainly based on the packet byte content, which needs to decode and parse data according to the specific DNS tunnel tool. These works require not only patience but also abundant professional knowledge and experience. Moreover, the volume and timeliness of these signatures affect the performance of the detection system.

In conclusion, signature-based detection with decent accuracy still has several drawbacks. The signature corresponding to the specific DNS tunnel tools or malicious activity data indicates low universality and

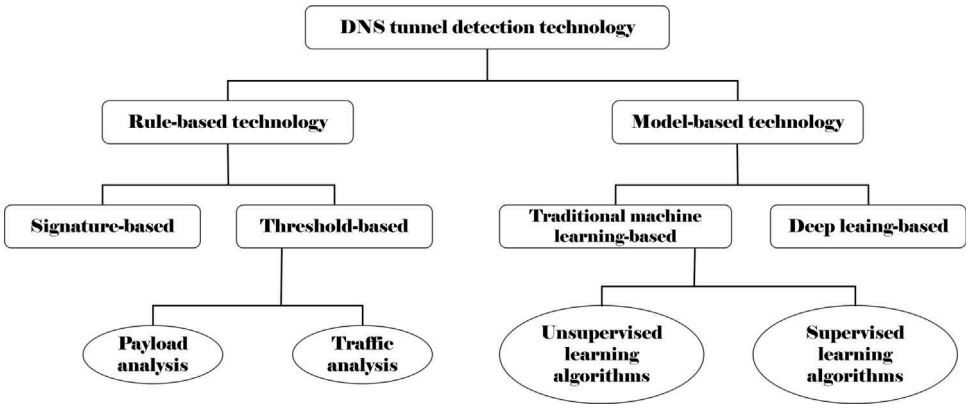

Fig. 6. The overview of DNS tunnel detection.

high resource consumption of human resources, time, etc. As shown in Table 4, the signatures mentioned above are only applied to NSTX and Iodine with an SSH connection, respectively. Unfortunately, they are of temporary validity; therefore, their updates and supplements, via professional analysis, are required. For example, [70] could reduce false negatives by completing its signature database, while not all malware would use SSH to establish connections that would disable these signatures. This adequately demonstrates the inflexibility of the signature , and the biggest defect is that only the DNS tunnel traffic known in the signature database can be identified. Despite the poor ability to respond to changes in tunnel traffic, the signature-based method can be easily applied to various types of networks, via common platforms, without complex configuration and hardware requirements. Table 5 presents a comparative summary of the literature on signature-based DNS tunnel detection methods. The × in the performance column of Table 5 means undefined in specific indicators or uncertain, while the ○ means various non-typical generic indicators.

4.1.2. The threshold-based methods

The threshold-based method involves detecting the DNS tunnel by comparing a specific threshold, which requires the quantitative analysis of the features of DNS tunnels to design a practical threshold value. The detection of DNS tunnels implements a comparison between the preset threshold and the value calculated by the traffic captured online. As opposed to the signature-based method, this method identifies DNS tunnels by comparing specific feature values. Its emphasis is to locate the threshold value of certain features that can distinguish between legitimate and tunnelling DNS traffic.

In addition to the direct quantitative analysis of relevant features, visualisation technologies are frequently applied for threshold determination or assisting detection. This technology presents multidimensional data in the form of visible graphs to help analyse the characteristics of large amounts of traffic data. Mohammed et al. [71] designed a system based on the parallel coordinates technique to visualizes and detect DNS tunnel traffic, consisting of a parser, an analyser, and a visualiser module. The analyser module filters out the right DNS packets captured on UDP port 53 by the parser module and calculates the number of DNS requests for each domain name according to the date. The top five results compared with the threshold set to 500 DNS requests to identify the suspicious domain were visualised with six parameters (source IP, destination IP, domain name, subdomain name, TTL, and time) by the visualiser. The visual graphs of DNS tunnels show four types of funnel-shaped attack modes that could be identified by the naked eyes. This method relies more on the visualisation technique and is unprofitable for real-time analysis and detection. Generally speaking, the preferable application of visualisation techniques is to assist in the analysis.

The selection of the threshold mainly relies on the quantitative analysis of specific features, generally from the perspectives of the payload and the overall traffic [61]. Next, we describe the relevant literature from these two perspectives.

- **The payload-based threshold analysis:** In payload analysis, thresholds usually focus on the statistic and domain name-relevant features of DNS packets. For instance, [72] calculated the hourly packet size histogram of the corresponding DNS packet types and identified the DNS tunnel using a cross-entropy-based packet size histogram anomaly detector and the inconsistent accumulation-based packet frequency detector. At present, researchers pay more attention to features of the domain name, such as the number of hostname characters utilised as a threshold by Jeffrey J. Guy [73]. An alert is set when the hostname, queried in packets, exceeds 52 or 27 characters in terms of that the length of the tunnel domain name should be longer than that of the legitimate domain name.
  Moreover, the characteristic features of the domain name could be utilised in threshold-based detection, as researched by the authors in [74]. They analysed distinctions between the legitimate domain name and the tunnel domain name from character frequency ranking and the changes in character frequency by ranking order. They then proposed research on developing the character frequency-based domain name fingerprint for DNS tunnel detection. In the same year, they designed an n-gram model combined with visualisation analysis for detection [75]. The standard fingerprint of the legitimate domain name is the feature vector space formed by the n-gram analysis of one million of the most popular domain name lists (AlexaTopSites) in 2009. The match value between the fingerprint of tha input DNS traffic and the standard fingerprint below the threshold indicates the existence of DNS tunnels. Furthermore, the detection performance of this system, utilising bigram analysis, would be better, because the bigram could provide more information in increasing computational and memory overhead. To this extent, this method emphasises the trade-off between performance and resource overhead costs.
  Similarly, from the perspective of character frequency analysis, the expected value of the bigram character frequency is regarded as the score of the domain name (ignoring the top-level domain) in [76]. According to the score distribution of the legitimate domain name and the tunnel domain name, the best score threshold was set to 0.0027, resulting in an accuracy of 98.74%. [77] is also based on the domain name score, combined with the isolation forest algorithm to obtain an abnormal score threshold and a scoring model.
  The similarity of the three papers, mentioned above [75–77], lies in the application of the domain name to generate a specific threshold, but the definition of the threshold is distinctive. [75] and [76] are mainly based on the character frequency analysis,

Table 4
Signature examples of specific DNS tunnel tools.

Object	Signatures	Description	Paper
Iodine	alert udp any any —>any 53 (content:"|01 00 00 01 00 00 00 00 00 01|"; offset:2; depth:10; content:"|00 00 29 10 00 00 00 80 00 00 00|"; msg:"covert iodine tunnel request"; threshold:type limit, track by_src, count 1, seconds 300; sid:5619500; rev: 1;)	The snort rules for the specific content encapsulated in the Iodine tunnel	[68]
	alert udp any 53 —>any any (content:"|84 00 00 01 00 01 00 00 00 00|"; offset:2; depth:10; content:"|00 00 0a 00 01|"; msg:"covert iodine tunnel response"; threshold: type limit, track by_src, count 1, seconds 300; sid:5619501; rev:1;)		
	alert udp any any —>any any (content:" - 45 10 -"; msg:"covert iodine tunnel request IP packet encapsulated"; offset:13; threshold:type threshold, track bysrc, count 10, seconds 5; sid: 5619500; rev: 1;)	The snort rule for the IP packets encapsulated in the Iodine tunnel	[69]
Dns2tcp	AAACCFNTSC0yLjAtT3BlblNTSF83LjJw MiBVYnVudHUtNHVidW50dTIuMg0	The specific strings for detecting SSH handshake encapsulated in Dns2tcp tunnels	[70]
NSTX	alert udp $EXTERNAL_NET any —>$HOME_NET 53 (msg:"Potential NSTX DNS Tunneling"; content:"|01 00|"; offset:2; within:4; content:"cT"; offset:12; depth:3; content:"|00 10 00 01|"; within:255; classtype:bad-unknown; sid:10002;)	The snort rule for NSTX's hardcoded value in the DNS header	[67]

Table 5
Summary of the signature-based detection methods.

Paper	Signature	Platform	Performance	Weakness
[68]	Specific contents encapsulated in the Iodine tunnel	Snort	×	Insensitive detection performance
[69]	The IP packets encapsulated in the Iodine tunnel	Snort	×	Vunerable to evasion technologies
[70]	The SSH handshake data encapsulated in the Dns2tcp tunnel	Bro Network Security Monitor	false positive <2% false negative ~10%	High false negative rate

which relies on the theory that the legitimate domain size obeys the Zipfs rule and the tunnel domain name obeys a random distribution. It is possible to successfully evade detection if the tunnel domain used for data transmission can be constructed with high-frequency characters. At present, it is incredibly difficult to implement, but generally has excellent performance.

- **The traffic-based threshold analysis:** In traffic analysis, thresholds usually focus on the entire traffic-related features instead of several packets. Such as [62] in 2020, the author regards the number of unique hostnames as the threshold for detection by Elasticsearch. The indicator of the DNS tunnel's existence is that the number of unique hostnames exceeds 300 with the non-existence of the domain in the whitelist. For the sake of data transmission and avoiding the failure of iterative queries for DNS cache, the number of unique hostnames in DNS tunnel traffic is greater than under the normal circumstances.

  Many studies have focused on the throughput of DNS traffic per domain for a DNS tunnel, which generates more data than the normal domain. Five DNS traffic bytes-based detectors were proposed in [78] using three approaches, which are based on a specific feature's ordinary threshold, the periodic change of its average value, and the distribution of its average value. One of these detectors for raw traffic alerts is when the number of traffic bytes exceeds 5000 , which lasts more than 55 s. Other detectors are used for the pre-processed data in terms of time series, implemented using different methods. The greatest strength of these detectors is that they can be applied to three scenarios of DNS tunnel attacks (data leakage, C2 command, and browsing HTTP) with decent performance. The entropy, that is more complex, could be the measurement of the throughput per domain, such as [79]. It proposes a base throughput metric, which is equivalent to the entropy of a probabilistic distribution query function generated by the domain's query count within a certain time interval. The DNS tunnel traffic does not always have a byte peak with a huge discrepancy in normal traffic because attackers can sacrifice part of the communication efficiency to reduce its throughput. In that case, the detection system's robustness would ordinarily be better under circumstances, such as a change of traffic or the entropy of query count in [78,79], rather than the simple throughput.

  More specifically, several researchers have focused on the volume of a particular resource record type of DNS packets. [80] designed a snort rule to alert when querying exceeded 20 TXT records in 60 s, and [17] also implemented detection based on TXT records. Strictly speaking, [17] without a specific threshold value is not a typical threshold-based detection method that detects payload distribution channels by DNS zone profiles, constructed by the resource record activities per domain. Nevertheless, its essential detection methods rely on the abnormality of the volume of TXT records per domain. Broadly speaking, this kind of approach is more targeted, but can be avoid. Because DNS tunnel tools usually can utilise various resource record types to distribute payloads, including TXT records, neither of the detection methods mentioned here could effectively detect the DNS tunnel generated by TUNs using CNAME records only.

  Additionally, thresholds could also be set in combination with various DNS information, such as [81] which regards the compression value of the total information content each client communicates via all its external DNS queries as the threshold. In

particular, Aiello et al. [82,83] combined principal component analysis (PCA) and mutual information (MI) to calculate a novel metric mi as the identification index, based on several statistical features. However, they found that the different circumstances of DNS server size or the traffic encapsulated in DNS tunnelling (e.g., P2P, SSH) would cause diverse manifestations of mi value. Hence, the threshold could only be determined based on the condition of the non-overlapping of the mi value between legitimate and malicious traffic, which is affected by many environmental factors. In other words, it indicates the poor flexibility and generality of this method.

Compared with the signature-based method, the threshold-based method is more flexible and universal. The signature searches whether there is a known specific DNS tunnel characteristic value in the traffic. The threshold is then used to identify the abnormality of traffic through a universal value. Thus, the threshold-based method can detect both known and unknown DNS tunnelling tools based on abnormal behaviour. The weakness of this method is the lack of flexibility and self-adaptability. Adjustment is required for various scenarios or environments to obtain a reliable threshold for detection. Table 6 presents a comparative summary of relevant papers on threshold-based DNS tunnel detection methods. The dataset column indicates the specific tools for constructing the DNS tunnels, and × represents uncertainty. The × in the performance column of Table 6 means undefined or uncertain specific indicators, while the ○ means various non-typical generic indicators.

4.2. The model-based detection

Model-based detection automatically generates identification rules based on several features via a model. Its emphasis lies in the training of the machine learning model. Machine learning algorithms can be subdivided into traditional machine learning and deep learning. The main difference between these two approaches lies in the method of feature extraction and algorithms. Traditional machine learning-based methods require specialists to manually extract critical features during the data processing stage via rich experience and professional knowledge. The deep learning-based methods can make full use of data structure and sequence information and extract critical features automatically.

4.2.1. The traditional machine learning-based methods

Machine learning is an algorithm model that automatically learns to optimise performance, with empirical data, to solve the target problem. The first step of traditional machine learning is feature engineering [84], selecting effective feature sets to process raw data, and choosing appropriate algorithms for model training. The DNS tunnel-related features are discussed in Section 3. Researchers would select some of these features from distinctive perspectives with some innovations to design a detection model.

In the DNS tunnel detection domain, common traditional machine learning algorithms are analysed with relevant papers from the perspectives of unsupervised learning and supervised learning.

- **Unsupervised learning algorithms of traditional machine learning:** In the unsupervised learning algorithm, the application of the isolation forest is relatively limited, which usually only needs to learn from benign samples. In [8], an accuracy of 95.07% and 98.49%, respectively, on the institute dataset and university campus dataset was achieved by the isolation forest with features of the requested domain name.

  The most commonly used unsupervised learning algorithm is the K-means clustering algorithm (K-means), which divides the sample set into K clusters, according to the distance between samples. In [16,38,55,85], the performance of K-means in DNS tunnel detection was studied, and impressive results were obtained. Therefore, [38] has a relatively low true-positive rate (91.68%). The data leak activity, distinguished from the DNS tunnel, is detected separately as other malicious traffic via the features of subdomains in [38]. Nonetheless, data leakage is a process of DNS tunnel activity. The controlled host transmits data to the master, while the DNS tunnels in [38] focus more on the stage when the master sends communications and instructions to the compromised host. The performance of data leakage detection achieves a high level of accuracy, 99.93%, through the logistic regression algorithm with statistical feature subdomains, which is far better than that of DNS tunnel detection which has a 91.68% true positive rate. The features utilised in DNS tunnel detection are statistical attributes of TXT record data contents, focusing on encoded data. In light of the DNS tunnel's principle, the response packets in a tunnel, which are for the transmission of control and command information, carry a relatively smaller amount of encoded data than the query packets for data leakage. Consequently, the small amount of encoded data may lead to insufficient information for detection, resulting in relatively poor system performance. The enhancement of its performance may rely on a more appropriate feature set.

  Furthermore, [55] showed that the one-class support vector machine (OCSVM) algorithm performs slightly better than the k-means algorithm in DNS tunnel detection under the scenario of a mobile phone network. The OCSVM algorithm, an extension of SVM, obtains an optimal hyperplane to achieve the maximum distance between the target data and the origin of coordinates [86]. Similar to the isolation forest algorithm, it only needs to train positive or negative data. In [55], the OCSVM applied four kernel functions to test, including linear, polynomial, radial basis function (RBF), and s-type kernel sigmoid. Among them, the polynomial obtains the best F1-value of 95%, and RBF can achieve 96% after adjusting gamma and "nu" parameters. In contrast, the F1-value of the K-means is no more than 70% under any scenario, with three types of initiation methods. The abysmal performance of the k-means model with a random value initiation indicates its insensitivity to unknown abnormal samples. One possible reason for this may be the small dataset generated by four clients, which contains fewer data and has a slight imbalance between malicious and legitimate DNS traffic.
- **Supervised learning algorithms of deep learning:** More researchers tend to use supervised learning for DNS tunnel detection, and [87] even proposed that some supervised learning algorithms are better than some unsupervised learning algorithms in this domain after a comparative study. For example, 99% accuracy was achieved using the logistic regression algorithm in 2006 [88].

  Among these common supervised learning algorithms, the most popular algorithm is the support vector machine (SVM) for DNS tunnel detection, which is an optimisation algorithm for solving convex quadratic programming. In [40], three algorithms (decision tree, SVM, and logistic regression) were applied to design a detection system composed of offline training and online testing stages. The offline training stage is to train a classification model via a specific algorithm, and then apply this model to detect the existence of DNS tunnels in the traffic captured by the sniffer at the on-line testing stage. The characteristic idea of [40] is that they extract four types of features based on the window size N (the number of request/response pairs in the window): time intervals, request packet size, sub-domain entropy, and resource record types. When the window size N is equal to 10, the SVM achieves an optimal accuracy of 99.77%, which is better than the other two algorithms. In addition, 17 found that multilabel SVM performs better than the multilabel Bayesian algorithm, and the

Table 6
Summary of the threshold-based detection methods.

Paper	Threshold	Dataset(malicious)	Performance	Weakness
[17]	A metric associated with the TXT records for DNS zone profiles (no specific value)	Malware Database using TXT records	×	• Limitation to the DNS tunnel based on TXT records • Vulnerable to the direct DNS tunnel due to the passive DNS technology
[71]	The number of requests for each domain name based on date and hours	×	×	The threshold has to be set appropriately for each network depending upon the typical traffic levels.
[72]	A metric associated with data throughput of domains	Iodine, Dnscat, Dns2tcp	○	• Update domain whitelist • The effect of DNS caching on detection effectiveness
[75]	A match value between the fingerprints of legitimate traffic and unknown traffic in domain names' character frequency	Iodine, tcp-over-DNS, Dns2tcp	×	The appropriate threshold needs to be set for different types of networks and different n-gram configurations.
[76]	The expect value of bigram character frequency	Dnscat	Accuracy 98.74% False positive 1.24%	Vulnerable to the high character frequency bigram of long DNS tunnels domain names
[62]	The amounts of unique hostname	Iodine, Dnscat2, malware DNSExfiltrator	×	Update domain whitelist
[78]	The number of bytes or the average number of bytes	Iodine	○	Alternative detectors require training of the relevant parameters for various scenarios.
[81]	The information content threshold based on the time resolution parameter	Ground truth DNS traffic (containing Iodine, a variant of Iodine, DNStunnel, NSTX, etc.)	×	Vunerable to situation with the combination of using multiple compromised clients and external name servers
[82]	A novel metric calculated by PCA and MI (no specific value)	Dns2tcp	×	The difficulty in defining a good separator under the case of SSH traffic encapsulated in DNS tunnels
[83]	Two novel metrics calculated by PCA and MI (no specific value)	Dns2tcp	False positive 0% False negative 0%	The threshold needs to be updated for various scenarios via one of metrics.

kernel SVM is better than the linear SVM. As the datasets and features of these papers are different, it is improper to compare these algorithms directly. However, the SVM algorithm has excellent performance in the DNS tunnel domain to some extent.

The typical ensemble learning algorithm of random forest (RF) is also popular in this domain [34,89]. [89] introduced a method of using the RF to detect DNS tunnels under a complex defence-in-depth enterprise environment. [34] defines the DNS flow session, formed by clustering DNS traffic in terms of 5-tuple information (source IP, source port, transport layer protocol, destination IP, and destination port). The model trained by the RF with the features of the DNS tunnel communication behaviour reaches an average true positive rate of 98.47%, but its false-positive rate is 5-7%. According to their analysis, such a high false detection rate is mainly since all hosts in the test environment have installed Qihoo 360's security products. This makes each host periodically send abnormal DNS packets to the IP of Qihoo 360's company, which is similar to communication behaviour of the DNS tunnel. Notwithstanding, a simple whitelist technology could reduce the false-positive rate to less than 0.01%. The innovation of [34] is the proposition of the DNS flow session, in that the DNS protocol based on the connectionless UDP has no definition of the flow session itself. With such a session defined, more available features could also be extracted, such as session duration. The DNS tunnel is generally long-term communication, whereas the normal process of the DNS query is short-term. Thus, it is a handy feature that can only be extracted under this condition.

Additionally, the random forest [90] can also be regarded as a weak classifier of a part of an ensemble classifier. On the basis of three elements, namely a set of classifiers, a combination rule, and the weights of each classifier, [91] constructs various ensemble classifiers using RF, k-nearest neighbour (KNN), and multilayer perceptron (MLP) in different ways. Two or three of them are combined with a majority or average rule and have diverse weights to form a robust classifier. The experimental results show that the combination rules and weights between weak classifiers have significant effects on the detection performance, and the ensemble classifier performs better than the single weak classifier. It is worth noting that an excess of weak classifiers is not conducive for detection performance. For example, [91] showed that the pairwise combination was superior to the three, among which the classifier composed of the RF and MLP with the combination of the majority is the best in their study. Moreover, because the RF itself is an ensemble classifier, a greater RF weight would result in a better classification effect.

In the early days, Aiello et al. [92] proposed a two-level classifier similar to the ensemble model on the foundation of the thought connecting to the first-level classifier constructed by the Bayes algorithm in [23]. Although these methods are not sufficient enough, the detection performance is also improved by further fusion of ensemble thoughts. Their research, together with other work mentioned above, fully illustrates the excellent performance of ensemble thought in DNS tunnel detection.

Based on the above, supervised learning algorithms have been widely used for DNS tunnel detection. The performance of the SVM algorithm is excellent and superior to the logistic regression algorithm, decision tree, Bayes, etc. Furthermore, the kernel-function-based SVM is better than the linear SVM. In unsupervised learning algorithms, the k-means is more popular, but its performance is not as good as

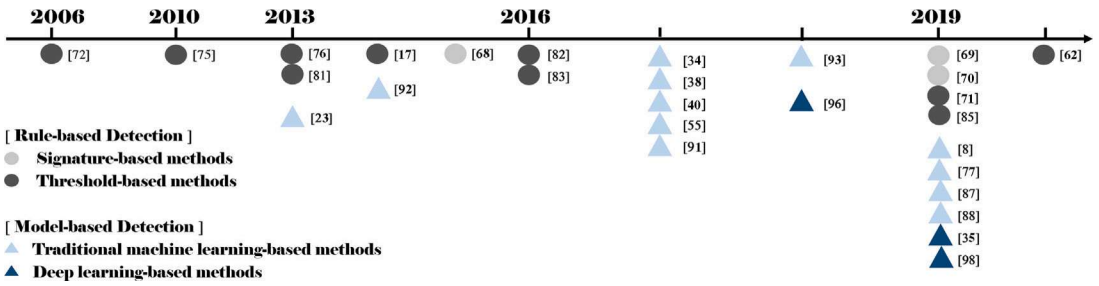

Fig. 7. The development of research relating to DNS tunnel detection.

that of the OCSVM under a mobile network scenario. The significance of ensemble learning is worth our attention as well. In addition to these machine learning algorithms, dataset and feature engineering are equally critical to model training. Hence, researchers must have a clear idea of the quantity and variety of DNS tunnel tools or malware utilised to generate datasets with different characteristics to perform a better job in feature engineering. Table 7 shows a comparative summary of the papers on the traditional machine learning-based method. The dataset column indicates the specific tools for constructing a DNS tunnel, and × indicates the uncertainty. × and √ represent the existence of two types of features: payload analysis and traffic analysis. The × in the performance column of Table 7 means undefined or uncertain specific indicators, while the ○ means various non-typical generic indicators.

4.2.2. The deep learning-based methods

Deep learning is a special machine learning algorithm that simulates human neurons to interpret data without artificial feature extraction [94]. Almost all of them can be considered as supervised learning. In DNS tunnel detection, the deep learning algorithms rarely applied are described as follows.

At the Defence Condition (DefCon) 17 Hacking Conference in 2009, the application of an artificial neural network (ANN) to identify the DNS tunnel is discussed [95]. Later, Aiello et al. [42] applied a neural network (NN), combined with other algorithms, for DNS tunnel detection. However, no research has focused on the performance of deep learning in this domain.

In the subsequent studies, researchers began to use a feed-forward neural network (FNN). For instance, [96] trained an FNN model using the first 512 bytes of the input DNS packet, and the final detection accuracy reached 99.96

The convolutional neural network (CNN) is the most popular deep learning algorithm for DNS tunnel detection. The character-level CNN may be limited to text content [97] for detection, such as the domain name. Therefore, [35] chooses the byte-level CNN that regards each field's byte as the unit to ensure the total utility of the entire DNS packet's content, the sequential, and structural information of packets. Before the model training, they convert each DNS packet into a fixed-length byte representation vector and truncate the overlength packets or fill the packets under length. After the embedding layer reduces the dimension of the byte encoded as a 257-dimensional vector, using the one-hot method, the pre-processed data are used to train the CNN model. Compared with three other algorithms, namely SVM, logistic regression, and neural network, the CNN model with an accuracy of 99.98% performs best. This indicator is significant in terms of demonstrating the superiority of CNN in DNS tunnel detection to some extent. The pre-processing of the dataset and the introduction of the embedding layer also play a critical role.

CNN was also applied in [98] to construct a system composed of the one-Model detection decision maker and the multi-model detection decision maker, combined with a dense neural network (DNN) and a recurrent neural network (RNN). They proposed four models: DNN, one-dimensional CNN (1D-CNN), RNN-LSTM, and RNN-GRU. The 1D-CNN model with the best Matthews correlation coefficient (MCC) value was selected as the one-model detection decision maker, and the best three models were selected as the multi-model detection decision maker based on the combination rule of a scoring algorithm. The final detection decision-maker consists of a one-model detection decision maker with 99.88% accuracy and a multi-model detection decision maker with 99.90% accuracy. As we can see, this method is a combination of deep learning and the thoughts of ensemble learning and of novelty and effectiveness, regardless of its hardware performance and system complexity requirements.

In summary, deep learning can perform extremely well in DNS tunnel detection, especially in CNN. The strength of the deep learning-based method is that it can utilise sequential and structural information as well as extract features automatically. However, this also means that the specific reasons affecting the performance of the deep learning model may be unknown and demand high-performance hardware. To the best of our knowledge, an excellent model can be obtained through parameter optimisation training. The dataset is critical for model training, which requires large amounts of data, and data pre-processing is also vital. Table 8 presents a comparative summary of the papers on the deep learning-based method.

4.3. Discussions

Fig. 7 shows that the development history, of research in this field, indicates that early DNS tunnel detection mainly focuses on rule-based methods. The rule-based methods usually combine with various platforms to manually set the rules according to the specific features. It could be easily deployed into various networks without complex configuration and high-performance hardware, which is more suitable for a stable network environment. The rule-based detection technology could be subdivided into signature-based and threshold-based methods. The former creates a signature database of the specific fixed data in known tunnel traffic for matching. The latter sets a threshold to distinguish between the normal traffic and tunnel traffic in terms of the universally applicable features. Both approaches require the artificial design of effective detection rules. Moreover, the signature-based method could only identify the DNS tunnel traffic with known signatures in the database. That makes the update and maintenance of the signature database very necessary. Similarly, the threshold-based method needs to adjust the exact threshold for different network environments or DNS tunnel activities' scenarios. Overall, the rule-based detection methods are of poor self-adaptability and inflexibility in coping with the changes in traffic or environment, and manual managements of detection rules are very complex and time-consuming. In that case, researchers increasingly tend to utilise machine learning training to detect DNS tunnels.

The core idea of model-based detection technology is to automatically generate rules for identifying DNS tunnels by training the algorithm models. It not only avoids the tedious and time-consuming rule design and system maintenance, but also has excellent self-adaptability and detection performance. Based on the appropriate dataset and features, it can deal with the attack of unknown DNS tunnel traffic. This

Table 7
Summary of traditional machine learning-based detection methods.

Paper	Algorithms	Features		Dataset(malicious)	Performance	Weakness
		Payload	Traffic			
[8]	Isolation Forest	√	×	An open source tool forked from DNS Exfiltration Toolkit (DET) project	Accuracy(Research Institute dataset) 95.07% Accuracy(University Campus dataset) 98.49%	High false positive rate
[23]	Bayes	√	×	Dns2tcp	○	Generality of the model
[34]	Random Forest	√	√	Dnscat2, DNSshell, cobalt strike	Average true negative 98.47% (DNS tunnels is defined negative)	Update domain whitelsit
[38]	Logistic Regression, K-means	√	×	Dnscat, Bernhardpos (malware)	•Aiming at data exfiltration(Logistic Regression): Accuracy 99.93% F1-Score 96% False positive 0.189% •Aiming at TXT records(K-means): True positive 91.68% False positive 0.4%	• Vunerable to different evasion technology • Detection of TXT records tunnelling is under-performance
[40]	Logistical Regression, Support Vector Machine (SVM), Decision Tree	√	√	Dns2tcp, Dnscat2, Iodine, OzymanDns	Best detection performance(SVM): Accuracy 99.96% Precision 99.98% Recall 99.93%	Vunerable to unknown DNS tunnels
[55]	K-means, One Class Support Vector Machine (OCSVM)	√	×	SlowDNS	Best detection performance(OCSVM): F1-Score 96%	• Unrepresentative for mobile network and small dataset lacking of sufficient variations • Unevaluated performance and scalability of its detection
[77]	Isolation Foest	√	√	Iodine, Dns2tcp, FrameworkPOS, Backdoor.Win32.Denis	×	Update domain whitelist
[87]	Random Forest, Gradient Boosting, AdaBoost, Bagging, Support Vector Classifier (SVC), Stochastic Gradient Descent (SGD)	√	√	a tool developed in Node.js to perform DNS tunnelling	○	• Dataset lacking of sufficient variations • Alternative algorithms need to be selected for various scenarios.
[88]	Logistic Regression	√	×	Dns-grind, Dns2tcp, Dnscapy, Dnscat2, Iodine, OzymanDns	Accuracy 99.99% Recall 99.99%	Vunerable to unknown DNS tunnels
[93]	SVM	√	×	Dns2tcp	○	Low accuracy of various protocols encapsulated in DNS tunnel detection
[91]	Random Forest, K-Nearest Neighbour (KNN), Multilayer Perceptron (MLP)	√	×	Iodine, OzymanDns, Dnscat2	○	Update ensemble model to catch up with potential changes in the network traffic behaviour
[92]	Bayes, KNN, Neural Networks (NN), SVM	√	×	Dns2tcp	○	Generality of the model

Table 8
Summary of deep learning-based detection methods.

Paper	Algorithms	Features	Dataset(malicious)	Performance	Weakness
[35]	Byte-level Convolutional Neural Networks (CNN)	DNS packet vector	Iodine, Dns2tcp, Dnscat2, OzymanDNS, ReverseDNShell	Accuracy 99.98% Precision 100% Recall 99.96% F1-Score 99.98%	Vunerable to DNS tunnels constructed by some DNS-based malware
[96]	Feed-forward neural network (FNN)	The first 512 bytes of DNS packets	Iodine	Accuracy 99.96% Precision 100% Recall 99.6%	Need improve detection generality
[98]	Dense Neural Network (DNN), Recurrent Neural Network (RNN), one-dimensional CNN (1D-CNN)	Integer encoded and paded subdomain	Iodine, Dns2tcp, Dnscat2, DNS reverse shell	• Accuracy (one-model detection decision make(1D-CNN)) 99.88% • Accuracy (multi-model detection decision maker(1D-CNN, RNN-LSTM, RNN-GRU)) 99.90%	• Lack of interpretability for deep learning model • High requirements of the hardware

type of technology is subdivided into traditional machine learning-based and deep learning-based methods. Compared with the traditional learning-based method, the deep learning-based method can automatically extract features and make use of sequential and structural information of data. Nonetheless, owing to the high requirements of the size of the dataset and the performance of hardware resources, traditional machine learning may be more applicable in most scenarios.

Reinforcement learning algorithms, without artificial feature extraction, also belong to machine learning algorithms. It describes and solves the problem of achieving maximum returns or implementing specific objectives through learning strategies during the process of interaction between the agent and the environment [99]. In DNS tunnel detection, the application of reinforcement learning is extremely limited, so it is not specifically mentioned as a type of detection technology in this study. In 2016, there was an attempt to detect a DNS tunnel's data leakage through the decision framework of the partially observable Markov decision process [100]. Furthermore, other special methods are worth investigating. For example, [101] designed a method to identify attacks against the DNS based on the exception of the DNS protocol's state transition, including DNS tunnels.

In addition to selecting DNS tunnel detection methods, the collection or simulation of datasets, including malicious and legitimate DNS traffic, is crucial. After eliminating possible malicious traffic, real legitimate DNS traffic is an excellent choice for the normal dataset. However, real DNS tunnel traffic is difficult to collect, which leads to a malicious dataset, usually consisting of simulation data created by DNS tunnel tools.

The concrete samples of the datasets depend on the input of the DNS tunnel detection system. After investigating the above literature, we find that input can be roughly divided into three categories: DNS flow, a single DNS packet, and domain names. DNS flow generally refers to DNS traffic aggregated by a domain within a certain time window, such as [77]. The DNS flow can also be grouped by quintuple information (source IP, source port, transport layer protocol, destination IP, destination port) proposed by [34]. As for the input of a single packet, the primary objective is to determine whether it is a request packet or a response packet, which is up to the detection system designer. The selection of features and methods influences the input. When the detection system focuses only on the payload-based features, the input could be a DNS packet (e.g. [35,55,96]), even the domain name (e.g. [8,38,76]). If there are traffic-based features, DNS flow is the system's input. Before these data are input into the system's core identification component, further processing and feature extraction are required to form feature vectors or be quantified into a value according to certain rules. The signature method typically enters a single packet and then directly checks the packet for a matching signature, usually without further action. At the same time, the deep learning method can also directly input data without feature engineering. In [35,96], they both take the bytes of a single packet of fixed length as the input. Finally, the identification component outputs the judgement results, which generally are the class of the input or the probability of various classes (e.g. [35]) and some alarm information (e.g. [8,34,38]). According to the analysis of input, the mainstream sample of the dataset in this field could be DNS flow, DNS packet, or domain name.

Furthermore, there is a great need to pay attention to the diversity and scale of the dataset, which would influence the generality and detection performance. Dataset diversity refers to various DNS tunnel traffic with different characteristics, depending on the categories of DNS tunnel tools. It can be seen in Table 6, Table 7, and Table 8 that several studies only applied one or two tools to create DNS tunnel traffic. This would reduce the performance and generality of detection systems. Therefore, the variety of mainstream DNS tunnel tools or DNS tunnels created by malware should be considered during the dataset collection phase. Next, the scale of the dataset is also a vital factor in detection performance. Suppose a detection system with outstanding performance is run on an extremely small-scale dataset. In this case, its performance metrics may degrade significantly when the detection system is applied to the real world or to a larger-scale dataset. In [77], the authors applied large-scale, high-quality DNS traffic logs (e.g., more than one million queries/hour) as the dataset, and a comparison between their method and two other methods [82,102], trained on a small-scale dataset, was conducted within this study. According to the experimental results, the performance degradation of the two methods was pronounced. Hence, the effectiveness and availability of such methods still needs to be tested in the real world. In conclusion, the scale of the dataset applied in this literature could be divided into three levels: large-scale datasets with more than a million data (e.g. [8,38,77]), a medium-scale dataset that ranges from a few hundred thousand to a million data (e.g. [35,40]), a small-scale dataset that usually ranges from hundreds of thousands to tens of thousands of data or created by few clients in a short time (e.g. [34,55,76,78,96]).

At the same time, the source of the dataset is also important as it affects the quality of the dataset and the performance of the method in the real world to some extent. Dataset sources refer to whether they are synthetic (lab generated) or collected in the wild. Datasets collected in the real world are generally more convincing regarding the effectiveness of a method than synthesised data. For the DNS tunnel traffic dataset, because it exposes very few datasets, it usually requires the experimenters to synthesise malicious data themselves. However, there are differences in the different synthetic environments. The insertion of a DNS tunnelling tool in the network boundary of real enterprises or campus environments to capture malicious traffic in real time [8,77,78,96] is closer to the real-world data than the construction of a small number of hosts in a closed internal network environment [34,40,55] for simulation synthesis. For normal DNS traffic, it is not difficult to collect real-world data. There are two types of collection methods of the literature studied in this paper. One is to collect publicly known benign domain names [8,76] based on Alexa (a website that publishes domain name rankings for free) or screen out the traffic queried on these benign domain names as normal DNS data [40]. The other is to manually clean the DNS data collected in the real world to form a benign dataset [38,77,96]. However, there are examples of the synthesis of normal flows in a simulated environment [34,55]. It can be seen that the source compositions of these datasets are very different, and all these factors will have a certain impact on the quality of the datasets. For the collection of data sets, it is better to use real world data because the ultimate purpose of these detection techniques is generally to be applied in the real world. If the gap between the detection techniques and real-world datasets is too large, the effectiveness of the methods may be significantly reduced or even ineffective.

Meanwhile, through the above discussion of dataset composition (DNS flow, a single DNS packet, and domain names), dataset size (large-scale, medium-scale, and small-scale dataset), and data sources (synthetic (lab-generated) or collected in the wild) in the research literature of this paper, it can be clearly seen that these methods cannot be directly compared, especially the performance indicators. The advantages and disadvantages of these methods should be analysed objectively from various aspects, such as datasets, features, and method principles, which is also the analysis principle followed in this paper.

Additionally, the DNS tunnel also has some evasion approaches. In the early days, [103] proposed two methods of DNS tunnel query for the normality of the query frequency distribution of tunnel traffic. Withal, it can only avoid detection based on the query frequency-related features, while payload analysis features, such as the domain name corresponding features, could easily invalidate it. A novel approach for encapsulating packets in the DNS tunnel was proposed in [104], which encapsulates data in the unused space of legitimate DNS packets, namely the relaxation space, instead of creating new DNS tunnel packets. Strictly speaking, DNS steganography is not exactly equivalent to DNS tunnelling according to the definition of the working principles. However, a DNS-based steganography technology proposed in [105] could indeed be regarded as DNS tunnelling, which aims

to encapsulate confidential information in the relaxation space of the DNS request packet, similar to [104]. This method constructs a query package with an answer and inserts the data after the correct answer in the answer field. Even though it is not permitted, it does not result in an invalid DNS query package. By sending the query packet directly to the controlled DNS server, confidential information transmission can be completed. For detection, DNS tunnelling packets, with confidential information in [105] are detectable because there is no answer to the DNS query packet under normal circumstances. Furthermore, a simple DNS proxy can prevent this circumvention. Unless the packets encapsulated by this method are sent directly to the controlled domain server, and its payload cannot survive, the proxy DNS server creates new packets to query the next server.

Furthermore, the low throughput heartbeat traffic in DNS tunnels is worthy of attention as well. Based on the working principle of the DNS protocol, the tunnel master disguised as a domain name server can only send data to the controlled host after receiving the DNS queries. However, a controlled host may not always have data for transfer. Consequently, the controlled host would periodically sends requests to the control end to maintain communication between them. These request packets constitute the heartbeat traffic of the DNS tunnel, and their data volume is less than the actual tunnel traffic of malicious activities. Similarly, the DNS tunnel may limit the amount and rate of its data transmission to avoid detection, resulting in a low throughput DNS tunnel as well. Even if the behavioural characteristics of the low-throughput traffic are less obvious, there still exists effective features to distinguish it from legitimate traffic, such as the volume of the heartbeat traffic within a certain time window [11].

5. Conclusion

This paper reviews various relevant DNS tunnel detection technologies so far. First, we introduce the essential background knowledge and DNS tunnel threats and briefly explain the working principles and classifications of DNS tunnel tools. The features related to the DNS tunnel behaviours are described in detail from the perspective of payload analysis and traffic analysis. Finally, DNS tunnel detection is classified into two categories according to the manual rule setting conditions: rule-based detection and model-based detection. The former is subdivided into signatures or thresholds for recognition, while the latter is subdivided into traditional machine learning-based and deep learning-based methods. To the best of our knowledge, this is the first comprehensive survey conducted in this field. It proposes a new classification to review a wide variety of DNS tunnel detection, instead of focusing on a single aspect of the DNS tunnel detection technologies, such as reviews [20,21]. Superior to other reviews, this paper analyses almost all the corresponding detection methods to present the whole technology development from 2016 to 2020, including advantages and disadvantages analyses.

To design a practical application of a DNS tunnel detection system, it is necessary to first investigate the system environment, including the performance of available hardware devices and network environment, limiting detection methods to some extent. For example, the deep learning method requires high-performance hardware devices. Some trained model methods or thresholds may need to be retrained and determined according to the network's different scale throughputs. Then, the background information can be combined with the performance requirements of the detection system to select and design the appropriate method, dataset, and information.

Common features are mainly divided into payload-based and traffic-based features. The payload-based features focus on analysing the payload's relevant characteristics by referring to several DNS packets. The traffic-based features aim to inquire about the overall DNS traffic in a specific time window, focusing more on the abnormal variation of traffic. Hence, the selection of the real-time detection system's features is almost always linked to payload, especially those related to domain names, making the system take the domain name as input (e.g. [8,76]). As for traffic-based features, several features may be closely related to the local network environment, such as the volume of DNS traffic to the IP address and the volume of DNS traffic to the domain. This type of feature indicates that the adjustment of detection systems using such features is essential for various network traffic scales. More information about the limitations and availability of DNS tunnel features is provided in Section 3. Moreover, for a low throughput DNS tunnel, although it works in the same way as the ordinary tunnel, it has the same tunnel characteristics. Its behaviour characteristics could not be different from legitimate traffic because of the low throughput malicious activity. Fortunately, there are still effective ways to distinguish it from legitimate traffic, such as focusing on the recent history of the domain rather than a short time window (e.g. [77]).

The selection of the method framework of the DNS tunnel detection system is a crucial link as well. The mainstream detection methods, including their advantages and disadvantages, are summarised in Section 4. Rule-based detection focuses on the manual analysis of significant features to design effective rules. Combined with various platforms, such methods can be easily deployed in existing networks. Nevertheless, this method cannot deal with a changeable network environment owing to poor self-adaptability, whether through signature or threshold. Model-based detection technology focuses on model training to form effective rules automatically, and it is flexible and has strong self-adaptability and university. These two types of technologies have different strengths and weaknesses, but both can perform well on the conditions of effective features and sufficient datasets.

Consequently, it emphasises choosing an appropriate DNS tunnel detection technology according to the actual diverse environment. Among these methods, it is difficult to build a robust and general detection system in isolation because of its poor adaptability and flexibility; therefore, it is often embedded in other systems as an aid module. Threshold-based methods and traditional machine learning-based methods are frequently used to design real-time detection systems, especially using the threshold-based method. As it identifies DNS tunnels by comparing a few specific feature values, it may require a longer running time for detection. As long as the appropriate features and quantification formula of the feature value are selected, the goal of real-time detection can be achieved. For instance, [76] designed a real-time detection system through the threshold of domain name scores which could process 26624 domain names per second with a high accuracy of 98.74%. Nonetheless, the performance of this system may suffer owing to its small-scale dataset in the real world. Finally, although the deep learning-based method has the advantages of automatic feature extraction, considering its high requirements for the size of the dataset and the performance of hardware resources, the traditional machine learning-based method may be more applicable in most scenarios.

Overall, the method, dataset, and features are the most vital factors of the detection system's performance. All three of these have an impact on the performance of the detection system and the scenarios (e.g., real-time detection, low throughput DNS tunnel detection) the system is good at, so there is a considerable challenge in comparing or designing a DNS tunnel detection system. It is difficult to judge the absolute quality of a DNS tunnel detection system using a specific criterion or to find a perfect detection system that meets any requirement. Therefore, when it comes to detecting DNS tunnels, the desirable idea is to combine the above general performance requirements and background environment to determine the appropriate method, dataset, and features step by step.

Moreover, future research could focus on two aspects: the DNS tunnel's heartbeat traffic and the protocols encapsulated in the DNS tunnel. The study of detecting the heartbeat traffic in DNS tunnels could eliminate the threat before it carries out malicious activities and avoids the situation where the detection performance may decline during the period of communication maintenance for the inactive DNS tunnel. The latter belongs to the network forensics technology, which analyses

the types of protocols under the awareness of the existence of DNS tunnels [106]. Some researchers have studied this, as in [107], which proposes a classification method limiting the encapsulated HTTP and FTP in the DNS tunnel. Instead of aiming at a single protocol, [108] is to identify pairwise mixing of three protocols, including simple mail transfer protocol (SMTP), SSH, and HTTP. Regardless of whether the detection is for a single protocol or mixed protocols encapsulated in DNS tunnels; this research still lacks exploration.

Meanwhile, it is worth noting that the emergence of DNS over transport layer security(DoT) [109] and DNS over the hyper text transfer protocol over securesocket layer(DoH) [110] has brought great challenges to DNS tunnel detection. These two emerging technologies are the encryption method to encapsulate DNS traffic into transport layer security(TLS) or hyper text transfer protocol over secure socket layer(HTTPS) protocols for secure transmission. In other words, DNS traffic can travel privately in TLS or HTTPS tunnels, which means that the existence of DNS tunnelling becomes the inner layer of encrypted double tunnelling. This leads to an increase in the concealment of DNS tunnelling and the difficulties in DNS tunnel detection. Therefore, with the popularisation of DoT and DoH in recent years, this new research direction is also in great need of attention and further research.

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.

Acknowledgements

This work is supported by the Key Research and Development Program of Sichuan province, China under Grant 2020YFG0076. All authors approved the version of the manuscript to be published.

References

[1] Rongfeng Zheng, Jiayong Liu, Weina Niu, Liang Liu, Kai Li, Shan Liao, Preprocessing method for encrypted traffic based on semisupervised clustering, Secur. Commun. Netw. 2020 (2020) 13.

[2] S. Zander, G. Armitage, P. Branch, A survey of covert channels and countermeasures in computer network protocols, IEEE Commun. Surv. Tut. 9 (3) (2007) 44–57.

[3] M. Dusi, M. Crotti, F. Gringoli, et al., Tunnel Hunter: Detecting application-layer tunnels with statistical fingerprinting, Comput. Netw. 53 (1) (2009) 81–97.

[4] Maarten Horenbeeck, Deception on the network: Thinking differently about covert channels, in; Proc. 7th Aust. Inf. Warf. Secur. Conf., http://dx.doi.org/10.4225/75/57a81f72aa0d6.

[5] Riyad Alshammari, A. Nur Zincir-Heywood, Can encrypted traffic be identified without port numbers, IP addresses and payload inspection? Comput. Netw. 55 (6) (2011) 1326–1350.

[6] B. Rajendran Sanjay, S.D. Pushparaj, DNS amplification & DNS tunneling attacks simulation, detection and mitigation approaches, in: Proc. Int. Conf. Inventive Comput. Technol, ICICT, 2020, http://dx.doi.org/10.1109/ICICT48043.2020.9112413.

[7] D. Dagon, M. Antonakakis, K. Day, et al., Recursive DNS architectures and vulnerability implications, in: Proc. Netw. Distrib. Syst. Secur. Symp., 2009.

[8] J. Ahmed, H.H. Gharakheili, Q. Raza, et al., Real-time detection of DNS exfiltration and tunneling from enterprise networks, in: Proc. IFIP/IEEE Symp. Integr. Netw. Serv. Manag., 2019.

[9] Yong Liu, XiangNan Gou, Research on application of feature analysis method in DNS tunnel detection, in: Proc. 2020 5th Annu. Int. Conf. Inf.N Syst. Artif. Intell., 2020, http://dx.doi.org/10.1088/1742-6596/1575/1/012069.

[10] Oskar Pearson, DNS Tunnel - through bastion hosts, 1998, [Online]. Available: https://seclists.org/bugtraq/1998/Apr/79.

[11] Greg Farnham, Detecting DNS tunneling, 2013, https://www.sans.org/reading-room/whitepapers/dns/detecting-dns-tunneling-34152.

[12] J. Li, B.K. Chandrasekhar, K.Y. Chan, Updating of malicious code patterns using public DNS servers, US 2012.

[13] Infoblox security assessment report, [Online]. Availabel: https://www.infoblox.com/wp-content/uploads/infoblox-security-assessment-report-2016q2.pdf.

[14] E. Skoudis, The six most dangerous new attack techniques and what is coming next? [Online]. Availabel: https://blogs.sans.org/pentesting/files/2012/03/RSA-2012-EXP-108-Skoudis-Ullrich.pdf.

[15] J.Q. Yang, L. Fang, Research on detection technologies of DNS-based covert channel, Mod. Comput. (20) (2013) 49–52, 56.

[16] C.J. Dietrich, C. Rossow, F.C. Freiling, et al., On botnets that use DNS for command and control, in: Proc. Eur. Conf. Comput. Netw. Def., http://dx.doi.org/10.1109/EC2ND.2011.16.

[17] A.M. Kara, H. Binsalleeh, M. Mannan, et al., Detection of malicious payload distribution channels in DNS, in: Proc. 1st IEEE Int. Conf. Commun., http://dx.doi.org/10.1109/ICC.2014.6883426.

[18] Josh Grunzweig, Mike Scott, Bryan Lee, Attacks use DNS requests as command and control mechanism, 2016, [Online]. Available: https://unit42.paloaltonetworks.com/unit42-new-wekby-attacks-use-dns-requests-as-comm{and}-and-control-mechanism/.

[19] Cian Lynch, Dimiter Andonov, Claudiu Teodorescu, Multigrain–point of sale attackers make an unhealthy addition to the pantry, 2016, [Online]. Available: https://www.fireeye.com/blog/threat-research/2016/04/multigrain_pointo.html.

[20] S. Yassine, J. Khalife, M. Chamoun, et al., A survey of DNS tunnelling detection techniques using machine learning, in: Proc. 1st Int. Conf. on Big Data and Cyber-Secur. Intell., vol. 2343, 2018, pp. 63–66.

[21] M. Sammour, B. Hussin, I. Othman, Comparative analysis for detecting DNS tunneling using machine learning techniques, Int. J. Appl. Eng. Res. 12 (22) (2017) 12762–12766.

[22] V. Nuojua, G. David, Timo Hämäläinen, DNS tunneling detection techniques – classification, and theoretical comparison in case of a real APT campaign, in: Proc. 17th Int. Conf. Next Gener. Teletraffic Wired/Wireless Adv. Networks Syst. 10th Conf. Internet Things Smart Spaces, 3rd Int. Workshop Nano-Scale Comput. Commun., 2017, http://dx.doi.org/10.1007/978-3-319-67380-6_26.

[23] M. Aiello, M. Mongelli, G. Papaleo, Basic classifiers for DNS tunneling detection, in: Proc. 18th IEEE Int. Symp. Comput. Commun., 2013, pp. 880–885.

[24] J. Klensin, RFC 3467 - role of the domain name system (DNS), 2003, [Online]. Available: https://tools.ietf.org/html/rfc3467.

[25] P.V. Mockapetris, RFC 1035 - Domain names - implementation and specification, 1987, [Online]. Available: https://tools.ietf.org/html/rfc1035.

[26] Li Li, Jiayong Liu, Peng Jia, Rongfeng Zheng, PSPAB: Privacy-preserving average procurement bidding system with double-spending checking, PloS One 15 (10) (2020) e0240548.

[27] S. Thomson, C. Huitema, V. Ksinant, M. Souis, RFC 3596 - DNS extensions to support IP version 6, 1995, [Online]. Available: https://tools.ietf.org/html/rfc3596.

[28] D. Eastlake, C. Kaufman, RFC 2535 - domain name system security extensions, 1997, [Online]. Available: https://tools.ietf.org/html/rfc2535.

[29] A. Gulbrandsen, P. Vixie, L. Esibov, RFC 2782 - A DNS RR for specifying the location of services (DNS SRV), 1996, [Online]. Available: https://tools.ietf.org/html/rfc2782.

[30] Torabi Sadegh, et al., Detecting internet abuse by analyzing passive DNS traffic: A survey of implemented systems, IEEE Commun. Surv. Tutor. 20 (4) (2018) 1.

[31] B. Zdrnja, N. Brownlee, D. Wessels, Passive monitoring of DNS anomalies, in: Proc. 4th GI Int. Conf. Detect. Intrusions Malware, and Vulnerability Assess, in: LNCS, vol. 4579, 2007, pp. 129–139.

[32] Li Li, Jiayong Liu, Peng Jia, SPCTR: Sealed auction-based procurement for closest pre-tender with range validation, Secur. Commun. Netw. 2020 (2020) 1–12.

[33] W. YongJie, L. JingJu, Principle and performance analysis of covert tunnel based on DNS protocol, Comput. Eng. 40 (7) (2014) 102–105.

[34] Youqiang Luo, Shengli Liu, Yan Meng, Dongying Wu, DNS tunnel Trojan detection method based on communication behaviour analysis, Zhejiang Daxue Xuebao (Gongxue Ban)/J. Zhejiang Univ. (Eng. Sci.) 51 (2017) 1780–1787.

[35] C. Liu, L. Dai, W. Cui, et al., A byte-level CNN method to detect DNS tunnels, in: Proc. 38th IEEE Int. Perform. Comput. Commun. Conf., http://dx.doi.org/10.1109/IPCCC47392.2019.8958714.

[36] P.V. Mockapetris, RFC 1034 - Domain names - concepts and facilities, 1987, [Online]. Available: https://tools.ietf.org/html/rfc1034.

[37] S. Josefsson, RFC 4648 - The base16, base32, and base64 data encodings, 2003, [Online]. Available: https://tools.ietf.org/html/rfc4648.

[38] A. Das, M.Y. Shen, M. Shashanka, et al., Detection of exfiltration and tunneling over DNS, in: Proc. 16th IEEE Int. Conf. Mach. Learning Appl., vol. 2017, 2017, pp. 737–742.

[39] S. Marchal, J. Francois, C. Wagner, et al., DNSSM: A large scale passive dns security monitoring framework, in: 2012 IEEE Net. Oper. Manage. Symp., vol. 131, no. 5, 2012, pp. 988–993.

[40] J. Liu, S. Li, Y. Zhang, et al., Detecting DNS tunnel through binary-classification based on behaviour features, in: Proc. 16th IEEE Int. Conf. Trust, Secur. and Privacy in Comput. Commun., 11th IEEE Int. Conf. on Big Data Sci. Eng. 14th IEEE Int. Conf. Embedded Software Syst., http://dx.doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.256.

[41] R. Zheng, J. Liu, K. Li, S. Liao, L. Liu, Detecting malicious TLS network traffic based on communication channel features, in; Proc. 8th IEEE Int. Conf. Inf. Commun. Net., http://dx.doi.org/10.1109/ICICN51133.2020.9205087.

[42] M. Aiello, A. Merlo, G. Papaleo, Performance assessment and analysis of DNS tunneling tools, Logic J. IGPL 21 (4) (2013) 592–602.

[43] A. Merlo, G. Papaleo, S. Veneziano, et al., A comparative performance evaluation of DNS tunneling tools, in: Proc. Comput. Sci., in: LNCS, vol. 6694, 2011, pp. 84–91.

[44] P. Vixie, RFC 6891 - Extension mechanisms for DNS (EDNS(0)), 1999, [Online]. Available: https://tools.ietf.org/html/rfc6891.

[45] NSTX, [Online]. Available: https://sourceforge.net/projects/nstx/.

[46] Dnscat2, [Online]. Available: https://github.com/iagox86/dnscat2.

[47] Iodine, [Online]. Available: https://code.kryo.se/iodine/.

[48] Lucas Nussbaum, Pierre Neyron, Olivier Richard, On robust covert channels inside DNS, in: Proc. 24th IFIP TC11 Int. Inf. Secur. Conf., vol. 297, 2009, pp. 51–62.

[49] OzymanDns, [Online]. Available: http://www.dnstunnel.de/.

[50] Heyoka, [Online]. Available: http://heyoka.sourceforge.net/.

[51] Dns2tcp, [Online]. Available: http://www.hsc.fr/ressources/outils/dns2tcp/index.html.en.

[52] Z. Wang, Combating malicious DNS tunnel, 2016, [Online]. Avaliable: arXiv:1605.01401.

[53] L. Bilge, E. Kirda, C. Kruegel, et al., EXPOSURE: Finding malicious domains using passive DNS analysis, in: Proc. Netw. Distrib. Syst. Secur. Symp., 2011.

[54] G. Zipf, Selected Studies of the Principle of Relative Frequency in Language, Harvard University Press, 2013, http://dx.doi.org/10.4159/harvard.9780674434929.

[55] V.T. Do, P. Engelstad, B. Feng, et al., Detection of DNS tunneling in mobile networks using machine learning, in: Proc. Int. Conf. Info. Sci. Appl., vol. 424, 2017, pp. 221–230.

[56] Bromberger Seth, DNS As a covert channel within protected networks, 2011, [Online]. Available: https://www.researchgate.net/profile/Seth_Bromberger/publication/319492986_DNS_as_a_Covert_Channel_Within_Protected_Networks/links/59aed9e2a6fdcca6542425c5/DNS-as-a-Covert-Channel-Within-Protected-Networks.pdf.

[57] P. Ren, J. Kristoff, B. Gooch, Visualizing DNS traffic, in: Proc. 3rd Workshop Visualization Comput. Secur., 2006, http://dx.doi.org/10.1145/1179576.1179582.

[58] D. Tatang, F. Quinkert, N. Dolecki, T. Holz, A study of newly observed hostnames and DNS tunneling in the wild, 2019, [Online]. Available: arXiv:1902.08454.

[59] D. Tatang, F. Quinkert, T. Holz, Below the radar: Spotting DNS tunnels in newly observed hostnames in the wild, in: Proc. 2019 APWG Symp. Electron. Crime Res., 2019, http://dx.doi.org/10.1109/eCrime47957.2019.9037595.

[60] Steffen Haas, Robin Sommer, Mathias Fischer, Zeek-osquery: Host-network correlation for advanced monitoring and intrusion detection, in: Proc. IFIP Advances in Inf. Commun. Technol., 2020, http://dx.doi.org/10.1007/978-3-030-58201-2_17.

[61] Steve Jaworski, Using splunk to detect DNS tunneling, 2016, [Online]. Available: https://www.sans.org/reading-room/whitepapers/dns/splunk-detect-dns-tunneling-37022.

[62] A.F. Sani, M.A. Setiawan, DNS Tunneling detection using elasticsearch, Proc. IOP Conf. Ser.: Mater. Sci. Engineering 722 (2020) 012064.

[63] Palo Alto Network, [Online]. Available: https://www.paloaltonetworks.com.

[64] S.A.R. Shah, B. Issac, Performance comparison of intrusion detection systems and application of machine learning to snort system, Future Gener. Comput. Syst. 80 (2017) 157–170.

[65] J. Vukalovic, D. Delija, Advanced persistent threats - detection and defense, in: Proc. Int. Conv. Inf. Commun. Technol. Electron. Microelectron., 2015, http://dx.doi.org/10.1109/MIPRO.2015.7160480.

[66] R. Bejtlich, Alert data network security monitoring using sguil, Comput. Secur. J. 20 (3) (2014) 15–32.

[67] Maarten Van Horenbeeck, Detection of DNS tunneling, [Online]. Available: https://www.daemon.be/maarten/dnstunnel.html#detect.

[68] S. Sheridan, A. Keane, Detection of DNS based covert channels, in: Proc. 14th European Conf. Inf. Warfare Security, vol. 2015, 2015, pp. 267–275.

[69] M. Al-Kasassbeh, T. Khairallah, Winning tactics with DNS tunnelling, Netw. Secur. 2019 (12) (2019) 12–19.

[70] Ghosh Tirthankar, El-Sheikh Eman, Jammal Wasseem, A multi-stage detection technique for DNS-tunneled botnets, in: Proc. 34th Int. Conf. Comput. Their Appl., vol. 58, 2019, pp. 137–143.

[71] Y.F. Mohammed, D.R. Thompson, Visualization of DNS tunneling attacks using parallel coordinates technique, in: Proc. 12th Int. Conf. Secur. Privacy Anonymity Comput. Commun. Storage, in: LNCS, vol. 11611, 2019, pp. 89–101.

[72] A. Karasaridis, K. Meier-Hellstern, D. Hoeflin, Detection of DNS anomalies using flow data analysis, in: Proc. GLOBECOM IEEE Global Telecommun. Conf., 2006, http://dx.doi.org/10.1109/GLOCOM.2006.280.

[73] J. Jeffrey, Guy dns part ii: visualization, 2009, [Online]. Available: http://armatum.com/blog/2009/dns-part-ii/.

[74] Kenton Born, David Gustafson, Detecting DNS tunnels using character frequency analysis, 2010, [Online]. Available: arXiv:1004.4358.

[75] K. Born, D. Gustafson, NgViz: Detecting DNS tunnels through N-gram visualization and quantitative analysis, in: Proc. ACM Int. Conf. Proc. Ser., 2010, http://dx.doi.org/10.1145/1852666.1852718.

[76] C. Qi, X. Chen, C. Xu, et al., A bigram based real time DNS tunnel detection approach, Procedia Comput. Sci. 17 (2013) 852–860.

[77] A. Nadler, A. Aminov, A. Shabtai, Detection of malicious and low throughput data exfiltration over the DNS protocol, Comput. Secur. 80 (2019) 36–53.

[78] W. Ellens, Piotr Żuraniewski, A. Sperotto, et al., Flow-based detection of DNS tunnels, in: Proc. 7th IFIP WG 6.6 Int. Conf. Autonomous Infrastructure, Manage. Secur., in: LNCS, vol. 7943, 2013, pp. 124–135.

[79] Michael Himbeault, A Novel Approach To Detecting Covert Dns Tunnels using Throughput Estimation, University of Manitoba, 2014, [Online]. Available: http://hdl.h{and}le.net/1993/23550.

[80] Jeffrey J. Guy, A study of DNS, 2009, [Online]. Available: http://armatum.com/blog/2009/a-study-of-dns/.

[81] V. Paxson, M. Christodorescu, M. Javed, et al., Practical comprehensive bounds on surreptitious communication over DNS, in: Proc. 22nd USENIX Secur. Symp., 2013, pp. 17–32.

[82] E. Cambiaso, M. Aiello, M. Mongelli, et al., Feature transformation and mutual information for DNS tunneling analysis, in: Proc. 8th Int. Conf. Ubiquitous Future Netw., vol. 2016, 2016, pp. 957–959.

[83] A. Maurizio, M. Maurizio, C. Enrico, et al., Profiling DNS tunneling attacks with PCA and mutual information, Logic J.. IGPL 24 (6) (2016) jzw056.

[84] Xun Tang, Juan Tang, AnMin Zhou, Research on community malicious comments detection based on a hybrid model of feature selection and random forest, Mod. Comput. (2020) http://dx.doi.org/10.3969/j.issn.1007-1423.2020.19.005.

[85] M. Aiello, M. Mongelli, M. Muselli, et al., Unsupervised learning and rule extraction for domain name server tunneling detection, Internet Technol. Lett. 2 (2) (2019) e85.

[86] B. Schlkopf, J.C. Platt, J. Shawe-Taylor, et al., Estimating support of a high-dimensional distribution, Neural Comput. 13 (7) (2001) 1443–1471.

[87] R. Preston, DNS tunneling detection with supervised learning, in: Proc. IEEE Int. Symp. Technol. Homel. Secur., 2019, http://dx.doi.org/10.1109/HST47167.2019.9032913.

[88] F.K. Wu, S.Y. Zhang, T.T. Yin, Clr: A classification of DNS tunnel based on logistic regression, in: Proc. 38th IEEE Int. Perform. Comput. Commun. Conf., 2019, http://dx.doi.org/10.1109/IPCCC47392.2019.8958731.

[89] A.L. Buczak, P.A. Hanke, G.J. Cancro, et al., Detection of tunnels in PCAP data by random forests, in: Proc. 11th Annu. Cyber Inf. Secur. Res. Conf., 2016, http://dx.doi.org/10.1145/2897795.2897804.

[90] Hualu Xu, Juan Tang, Jiayong Liu, Research on random forest-based detection of weibo zombie account, Mod. Comput. (30) (2020) 16–20.

[91] S. Shafieian, D. Smith, M. Zulkernine, Detecting DNS tunneling using ensemble learning, in: Proc. 11th Int. Conf. Netw. Syst. Secur., in: LNCS, vol. 10394, 2017, pp. 112–127.

[92] M. Aiello, M. Mongelli, G. Papaleo, Supervised learning approaches with majority voting for DNS tunneling detection, in: Proc. Int. Joint Conf. SOCO, CISIS, ICEUTE, vol. 299, 2014, pp. 463–472.

[93] A. Ahmed, A. Haleh, DNS Tunneling detection method based on multilabel support vector machine, Secur. Commun. Netw. 2018 (2018) 1–9.

[94] S. Liao, J. Liu, X. Xiao, et al., Modified gradient neural networks for solving the time-varying sylvester equation with adaptive coefficients and elimination of matrix inversion, Neurocomputing 379 (Feb. 28) (2020) 1–11.

[95] Hind. J., Catching dns tunnels with a.i, in: Proc. Def. Cond. 17 Hacking Conf., 2009, [Online]. Available: https://defcon.org/images/defcon-17/dc-17-presentations/defcon-17-jhind-dns_tunnels_with_ai.pdf.

[96] C.M. Lai, B.C. Huang, S.Y. Huang, et al., Detection of DNS tunneling by feature-free mechanism, in: Proc. 2018 IEEE Conf. Dependable Secure Comput., 2018, http://dx.doi.org/10.1109/DESEC.2018.8625166.

[97] Y. Chen, R. Zheng, A. Zhou, et al., Automatic detection of pornographic and gambling websites based on visual and textual content using a decision mechanism, Sensors 20 (14) (2020) 1–21.

[98] J. Zhang, L. Yang, S. Yu, et al., A DNS tunneling detection method based on deep learning models to prevent data exfiltration, in: Proc. 13th Int. Conf. Netw. Syst. Secur., in: LNCS, vol. 11928, 2019, pp. 520–535.

[99] Hui Fang, Danning Zhang, Yiheng Shu, et al., Deep learning for sequential recommendation, ACM Trans. Inf. Syst. (2020) http://dx.doi.org/10.1145/3426723.

[100] S.M.M. Carthy, A. Sinha, M. Tambe, et al., Data exfiltration detection and prevention: Virtually distributed POMDPs for practically safer networks, in: Proc. 7th Int. Conf. Decis. Game Theory Secur., LNCS, vol. 9996, 2016, pp. 39–61.

[101] P. Satam, H. Alipour, Youssif Al-Nashif, Hariri Salim, Anomaly behaviour analysis of DNS protocol, J. Internet Serv. Inf. Secur. (2015) http://dx.doi.org/10.22667/JISIS.2015.11.31.085.

[102] Irvin Homem, Panagiotis Papapetrou, Spyridon Dosis, Entropy-based prediction of network protocols in the forensic analysis of DNS tunnels, 2017, [Online]. Available: arXiv:1709.06363.

[103] P. Butler, K. Xu, D. Yao, Quantitatively analyzing stealthy communication channels, in: Proc. Int. 9th Conf. Appl. Cryptography Netw. Secur., in: LNCS, vol. 6715, 2011, pp. 238–254.
[104] Kenton Born, PSUDP: A passive approach to network-wide covert communication, black hat USA, 2010, [Online]. Available: https://media.blackhat.com/bh-us-10/whitepapers/Born/BlackHat-USA-2010-Born-psudp-Passive-Network-Covert-Communication-wp.pdf.
[105] Szczypiorski Krzysztof, Michał Drzymała, Marek Łukasz Urbański, Network steganography in the DNS protocol, Int. J. Electron. Telecommun. 62 (4) (2016) 343–346.
[106] A. Berg, D. Forsberg, Identifying DNS-tunneled traffic with predictive models, 2019, [Online]. Available: arXiv:1906.11246.
[107] I. Homem, P. Papapetrou, S. Dosis, Information-entropy-based DNS tunnel prediction, in: Proc. 14th IFIP WG 11.9 Int. Conf. Digit. Forensics, vol. 532, 2018, pp. 127–140.
[108] H. Bai, G. Liu, J. Zhai, et al., Refined identification of hybrid traffic in DNS tunnels based on regression analysis, ETRI J. (2020) http://dx.doi.org/10.4218/etrij.2019-0299.
[109] Rebekah Houser, Zhou Li, Chase Cotton, Haining Wang, An investigation on information leakage of DNS over TLS, in: The 15th Int. Conf. Emerging Networking Exp. Technol., 2019, http://dx.doi.org/10.1145/3359989.3365429.
[110] Dmitrii Vekshin, Karel Hynek, Tomas Cejka, DoH Insight: detecting DNS over HTTPS by machine learning, in: 15th Int. Conf. Availability Reliab. Secur., 2020, http://dx.doi.org/10.1145/3407023.3409192.

Yue Wang is currently pursuing her M.S. degree in School of Cyber Science and Engineering, Sichuan University, Sichuan, China. Her current research interests include cyber security, network traffic analysis, and natural language processing.
E-mail: zicooo@qq.com

Anmin Zhou received the B.Eng. degree from the Northwest Institute of Telecommunication Engineering, Xi'an, China, in 1984. He is currently a Research Fellow in the School of Cyber Science and Engineering, Sichuan University, China. His current research interests include malicious detection, network security, system security, and artificial intelligence.

Shan Liao received her master degree in software engineering in 2012 from Guangdong University of Technology, Guangzhou, China. She is currently pursuing her Ph.D. degree in School of Cyber Science and Engineering, Sichuan University. Her current research interests include information mining, neural network algorithm, and signal processing.
E-mail: liaoe1110@163.com

Rongfeng Zheng received the M.S. degree from Sichuan University, China, in 2016. He is currently pursuing the Ph.D. degree in the Electronic Information College of Sichuan University since 2016. His primary research interests include cyber security, threat intelligence, and Internet of Things (IoT) security.

Rong Hu is currently pursuing her M.S. degree in School of Electrical Engineering, Sichuan University, Sichuan, China. Her current research interests include reactive power optimisation and smart grid.

Lei Zhang received the M.S. and Ph.D. degrees in computer science and technology from Sichuan University, Chengdu, Sichuan, China, in 2010 and 2015 separately. He is currently an assistant researcher in School of Cyber Science and Engineering, Sichuan University. His research interests include malicious detection, machine learning and mobile security.
E-mail: zhanglei2018@scu.edu.cn