Computers & Security 132 (2023) 103357

Contents lists available at ScienceDirect

Computers & Security

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

DNS tunnelling detection by fusing encoding feature and behavioral feature

Yu Tu[a], Shuang Liu[a,b,*], Qian Sun[c]

[a] College of Intelligence and Computing, Tianjin University, Tianjin 300072, China
[b] Key Laboratory of Safety-Critical Software (Nanjing University of Aeronautics and Astronautics), Ministry of Industry and Information Technology, Nanjing, 211106, China
[c] School of Electronic and Information Engineering, Nanjing University of Information Science and Technology, Nanjing 210044, China

ARTICLE INFO

Article history:
Received 31 January 2023
Revised 16 May 2023
Accepted 18 June 2023
Available online 19 June 2023

Keywords:
DNS
Tunnelling detection
Encoding feature
Behavioral feature
Real-time detection

ABSTRACT

For a long time, DNS tunnels have played a pivotal role in advanced persistent threat (APT) attacks, posing a threat to the real-world network environment. Existing approaches either use single packets or grouped packets as the basic unit for detecting DNS tunnelling traffic. Currently, single-packet-based methods usually focus on the encoding features of the DNS packet payload, which results in a high false alarm rate in real-world environments. Approaches using grouped-packets as the basis detection unit are difficult to apply to real-time inspection environments and cannot perform a finer-grained inspection of packets. To address these issues, we propose a DNS tunnelling detection method that uses a combination of encoding and behavioral features on DNS query & response pairs (Q&R). Q&R pairs achieve good balance between the information conveyed and the detection time required. We then train a powerful XGBoost classifier to fuse the features and conduct the classification of benign DNS traffic and DNS tunnelling traffic. We evaluate our approach with an authoritative DNS dataset (the base dataset) and a generalized test dataset that we curated with real-world attack simulations and collections of secure enterprise traffic. Our method achieves 99.68% accuracy on the base dataset and 98.82% accuracy on the generalized test dataset, which is better than the compared state-of-art methods. The efficiency is 66.77% higher than the baseline grouped-packets-based method. We also test our method in a real-world network environment and detect 7 malicious DNS tunnelling events.

1. Introduction

Domain name server (DNS) tunnelling is a type of cyber attack technique, which encodes the malicious data payloads that may be added to an attacked DNS server to remotely control other servers. DNS tunnelling has been used as a core technique in the command&control phase of advanced persistent threat (APT) (Yunakovsky and Pomerantsev, 2017), in which attackers use DNS tunnels to build a covert channel to bypass firewalls and avoid detection.

For instance, in July 2020, the United Kingdoms National Cyber Security Centre (NCSC) and Canadas Communications Security Establishment (CSE) released a joint report (Centre, 2020), stating that the "APT29" group used the "WellMess" malware family, which adopted the DNS tunnelling technology to maintain network connections and launch attacks against organizations related to new coronavirus research and vaccine development. The growing number of attacks being successfully conducted also demonstrate that the DNS tunnelling technique is actively adopted by various real-world attacks, which threaten the Cyber-Security ecology. It has been reported that near half of the enterprise networks were assessed as existing DNS tunnels with high probability (Infoblox, 2016).

Several research approaches have been proposed to detect DNS tunnels, and they can mainly be categorized into two classes based on the granularity of the payload packets. The first class of methods (Ahmed et al., 2019; Chen et al., 2021; D'Angelo et al., 2022; Liu et al., 2019) use a single DNS packet as the detection unit and conducts real-time monitoring of the DNS tunnelling through exploring encoding methods of the single packet's payload. Those approaches suffer from low detection accuracy, especially in real-world traffic, due to the simple features that they adopt. The second class of methods (Ishikura et al., 2021; Saeli et al., 2020; Yang et al., 2020) use grouped DNS packets as the detection unit and aims to improve the detection accuracy by extracting traffic fea-

* Corresponding author.
E-mail address: shuang.liu@tju.edu.cn (S. Liu).

https://doi.org/10.1016/j.cose.2023.103357
0167-4048/

tures from a more macroscopic perspective. Those methods perform statistical analysis of traffic in a fixed window and show better performance than single packet detection methods on their offline datasets, yet suffer from efficiency issues when deployed online.

To mitigate the deficiencies of both class of methods, we propose to use query-response (Q&R) pairs, which are the smallest units containing both encoding content and traffic anomalies in the DNS tunnel, as the basic detection unit. Compared with grouped-packets-based detection methods, our approach based on Q&R pairs consumes less storage resources and does not need to consume time windows to group DNS packets, which greatly improves the detection efficiency. Compared with the single-packet-based detection methods, our approach based on Q&R pairs can extract more feature dimension, which greatly improves the accuracy and generalization ability. In our method, the encoding features are extracted with the attention-mechanism encoder, and the statistical DNS traffic features are manually extracted features. Those two types of features complement each other and we design an ensemble learning model to fuse those features for better detection performance.

Following existing works, we conduct our evaluation with a widely adopted open-source dataset (Chen et al., 2021) and compare our approach with three state-of-the-art approaches (Chen et al., 2021; D'Angelo et al., 2022; Yang et al., 2020). To further validate the detection capability of different methods under real attacks, we also construct a generalization dataset by crawling the traffics generated from actual penetration tests of seven DNS tunnelling tools.Our method achieves 99.68% and 98.82% on the F1 score on the base dataset and the generalization test set and outperforms state-of-the-art approaches by 1.44% and 10.02%, respectively. We also conduct an online detection in a campus environment, and successfully detect 7 DNS tunneling attacks.

Novelty and Contribution We made the following contributions in this work.

1. To solve the problem of low detection accuracy of single packet-based detection methods and the poor efficiency of group packets-based detection methods, we propose to use Q&R pair as the detection unit for the first time. Compared with single packet detection, DNS Q&R pairs contain richer feature dimensions, which can improve the accuracy. At the same time, compared with the existing grouped packets detection methods, Q&R pair has low construction overhead, which can effectively improve the efficiency of packet detection.
2. To construct more representative features, we introduce a multi-head attention mechanism for encoding feature extraction in DNS tunnelling traffic detection. Moreover, we fuse the encoding feature with statistical feature with an ensemble learning model for better detection performance.
3. To thoroughly evaluate our approach, we construct a generalization test dataset that matches the real attack environment by simulating attacks. We experimentally evaluate our method on the base dataset and the generalization test dataset. The results show that the detection accuracy and generalization ability of our method is better than the baseline methods. Moreover, we conduct an online test in real campus environment and our method successfully detect 7 DNS tunneling attacks.

The remainder of this paper is organized as follows. In Section 2, we introduce the background knowledge related to DNS tunnels. In Section 3, we introduce the details of our method. We report our experimental results in Section 4. Section 5 summarizes the related works, and finally, we conclude our approach in Section 6.

2. Background

In this section, we provide some background knowledge of DNS tunnels, including the format of DNS packets (Section 2.1), the process of tunnelling communication using DNS packets (Section 2.2), and the DNS payload encoding (Section 2.3).

2.1. DNS Packet format

The DNS protocol uses the packet to establish a mapping relationship between the domain name and the internet protocol (IP). Fig. 1 shows the format of a standard DNS packet. Each DNS packet contains a fixed length header and the DNS payloads section of various lengths. **The Header of a DNS packet.** The header of the DNS packet consists of six 2-byte fields. The ID is a 2-byte identifier that can be used to identify the correspondence between DNS response packets and DNS query packets. The Flags field contains various flags of DNS response and query methods. The last 8 bytes of the DNS header are counters of the number of entries in the question section (QDCOUNT), the number of resource records in the answer section (NSCOUNT), the number of authority resources (ANCOUNT), and the number of resource records in the additional records section (ARCOUNT). Since most of the characters in the DNS header are used for identification, the characters in those fields do not change frequently. **The Payload of a DNS packet.** The payload of the DNS packet contains the question section and the resource record (RRs) section which is further partitioned into the answer area, the authority area and the additional area. The question section is used to carry the query to be delivered, and it has the query name, query type, and query class. The query name is usually the domain name to be queried, or sometimes the IP address used for reverse queries. The query type can be "A", "AAAA" or "CNAME", etc. Different query types are used to instruct the DNS response message to respond with different types of resource data. The query class specifies the class, e.g., "IN" for the Internet, of the query. The resource record section is used in DNS packets to transmit server response data. Since both the question and resource-record sections are writable, and DNS packets are rarely blocked by firewalls, the DNS protocol is often used to build tunnels (Raman et al., 2012).

2.2. The DNS tunnelling process

DNS tunnelling is a type of tunnel technique based on the DNS protocol, and it utilizes the DNS query process and encapsulates the data in the DNS query/response package to build a proprietary tunnel for transmission communication between the controlled host and the attacker (Merlo et al., 2011). APT organizations generally control user machines through back-doors reserved in the software or upload remote control software. After the control software is activated, the hacker will use its own fake domain name server to transmit information to the user through DNS tunnelling. Fig. 2 shows an example of the communication process between the controlled host and the attacker. We provide a stepwise explanation of the process, and we assume that the domain name "attacker.com" is shared in this example. **[Step 1]** The controlled host first converts the data to be transferred to contents of specific encoding format, such as Base32, Base64, and Base128. Then it wraps the content to be transmitted in a DNS query packet and initiates a normal DNS resolution query. **[Step 2–4]** Following the normal process of resolving a subdomain, the local name server iteratively queries the root name server, the top-level name server (for domain "com"), and the disguised name server (for domain "attacker.com"). **[Step 5]** Then, the disguised name server obtains the message by decoding the query packet and replies with an encoded DNS response containing the command to the controlled

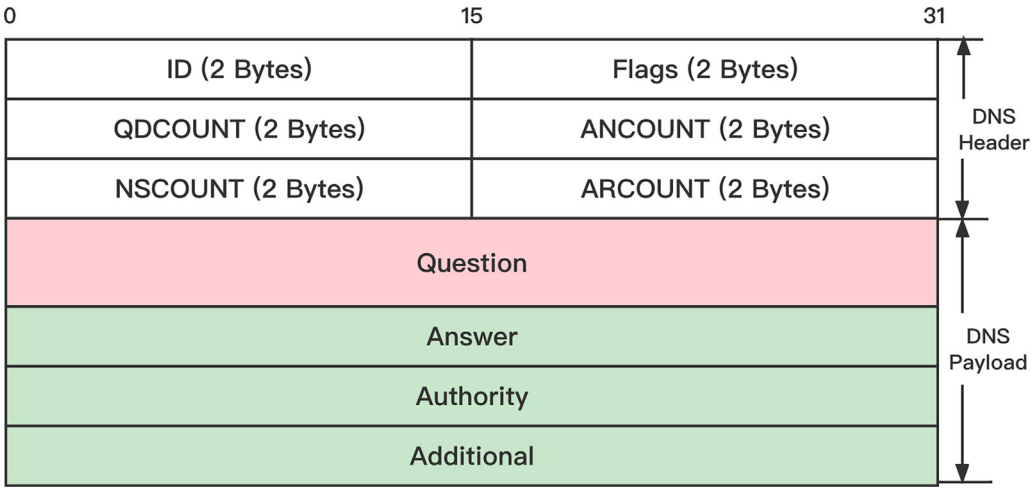

Fig. 1. The standard format of a DNS packet.

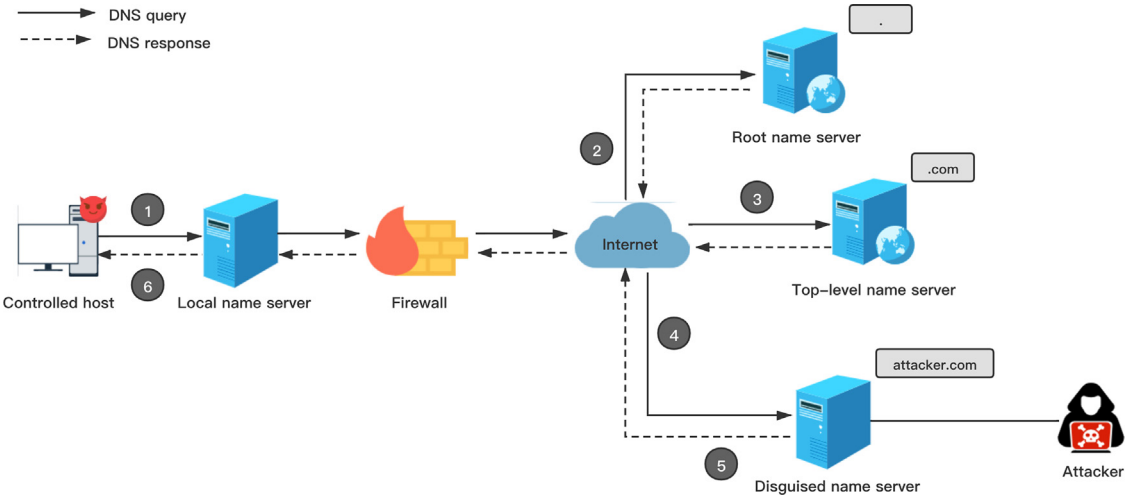

Fig. 2. DNS tunnel communication process between the controlled host and the attacker.

host. **[Step 6]** After repeating the process of obtaining a new command and sending a reply to the command, the controlled host eventually interacts with the attacker in the same manner.

2.3. Encoding of DNS tunnelling payload

The DNS tunnel usually utilizes an encoder, a decoder and a transport channel (Zander et al., 2007). The encoder and decoder are used to encode the DNS data when it is sent and decode the encoded DNS data when it is received. The transmission channel is a channel used to transmit the DNS data. To hide the transmission of information and meet the request for comments (Group, 1987) of the DNS in different scenarios, different encoding methods are usually proposed. We summarize the encoding methods of the DNS tunnel in Table 1, in which we categorize the methods based on whether they are used in real-life DNS tunnel attacks or open-source DNS tunnel tools.

Base32/Base64 are favored by the real cyber attack. Wekby (Grunzweig, 2016) generates a random 10-byte long alphanumeric header and encodes the rest of the data by Base32 to hide the instructions. To better escape detection, Turla (Kaspersky, 2017) utilizes Base64 to wrap the convert data into the IPV6 DNS response packet (YUNAKOVSKY and POMERANTSEV, 2018). Multigrain (Cian Lynch and Teodorescu, 2016) utilizes a customized Base32 encoding method to encode the DNS data.

Table 1
Summaries of DNS tunnel encoding methods .

Category	Name	Encoding methods
Real attack	Wekby	Base32
	Turla	Base64
	Multigrain	Custom Base32
DNS tunnel tool	Dnscapy	Txtfy
	Iodine	Base32, Base64, Base128
	Dnscat2	Salsa20
	DET	AES256
	Ozymandns	Base32, Base64

Among the DNS tunnel tools. Dnscapy (Bienaimé and Mazon, 2011), a lightweight DNS tunnel tool, makes an encoding method Txtfy for the TXT records. In Txtfy encoding method, every 255 characters will be divided into a character segment, and the length of each character segment will be converted into a new character through `Unicode` and added to the head of the current character segments. By concatenating these character segments, a new Txtfy-encoded string is obtained. Through the encoding method of Txtfy, the data can be decoded faster in the case of data confusion. Iodine (Ekman, 2021), a highly customizable software covers IPv4 data through a DNS server. It encodes information

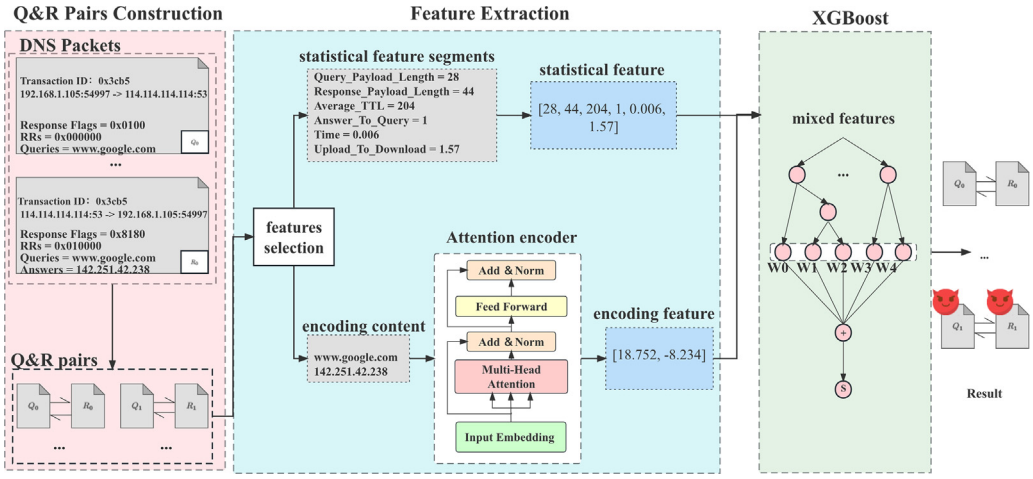

Fig. 3. Overview of the DNS tunnelling detection method.

into different types of DNS services via Base32 or Base64 encoding. Dnscat2 (Ron, 2021) is designed to create an encrypted command&control channel over the DNS protocol. It can automatically encode the data into the DNS protocol via Salsa20. DET (Jay Carlson and Stalmans, 2017), is a proof of concept tool to perform Data ex-filtration using either single or multiple channel(s) at the same time. Ozymandns (janprunk, 2021) is an early DNS tunnel tool with data requests encoded in Base32 and responses encoded in Base64. These DNS tunnel tools appear in the majority of datasets for DNS tunnel detection. However, due to the unfamiliarity of data builders with network attack operations, there is no network attack payload included in the open dataset. We use these popular DNS tunnelling tools to construct generalization test datasets that we explicitly sample for attacks.

3. Method

In this section, we propose a DNS tunnelling detection method, which utilizes the Q&R pair as the basic unit for detection, and fuses the encoding feature and the traffic anomalies feature to enhance the detection accuracy.

3.1. Overview

The overview of our method is shown in Fig. 3. Our system consists of three components, i.e., Q&R pairs construction, feature extraction, and classification.

Q&R pair construction. The first component is Q&R pair construction, which constructs Q&R pairs from DNS traffic. We divide the DNS traffic by the network quintuple, which consists of the source IP, destination IP and the protocol number of the IP layer; the source port and the destination port of the transport layer. Then, we use the DNS transaction ID and DNS response identification number reserved in the application layer to divide DNS Q&R pairs. The DNS Q&R pair is obtained from the diverted DNS traffic by using the DNS transaction ID. And for the paired DNS traffic, the DNS response identification number is used to divide the query part and the response part. It's worth mentioning that our DNS Q&R pairs are constructed in such a way that they are independent of DNS query type, such that they are applicable to any DNS traffic. In the Table 2, we show some common DNS query types and their content segments in DNS Q&R pair. **feature extraction.** We extract two types of features, i.e., the encoding feature and the statistical traffic feature, from the DNS Q&R pairs. The encoding feature captures the textual contents of Q&R pairs, while the statistical feature captures the length, size and time interval of the DNS payloads. We fuse both types of features as the representation of Q&R pairs. **Classification.** The last component is a classification model. We adopt a gradient boosting tree algorithm based on decision trees, due to its ability to handle data in a non-linear fashion, perform feature importance assessment, and the ease of interpretation of the resulting models.

3.2. Encoding feature extractor

The encoding content of normal DNS Q&R pairs can be regarded as semantic sequences, while the encoding content in DNS tunnel is covert data with different encoding methods, which are more similar to random sequences. Considering that the recurrent neural network (RNN) is good at processing serialized input, we adopt the RNN class model for encoding feature extraction. We also adopt the Attention mechanism to effectively improve the problem of gradient vanishing (due to the continuous recursion of the network loop).

We first conduct noise character removal and word-to-index mapping in the pre-processing step (shown in Algorithm 1) to

Algorithm 1: pre-processing of Q&R pair content

Input: *content* // content of Q&R pair;
word2index // word to index map;
Output: *result* // pre-processed content;

```
for query in content.query do
    if query.count('.') >= 2 then
        query.remove(TLD);
        query.remove(2LD);
    end
    result += query;
end
// Use tab to split queries and answers
result += Tab;
for answer in content.answer do
    answer = answer.remove(':');
    result += answer;
end
result.remove('.');
result.lower();
for char in result do
    char = char.word2index();
end
```

Table 2
Different types of DNS Q&R content .

Type	Query content	Response content
A	google.com	142.251.42.238
AAAA	www.google.com	2a03:2880:f12d:83:face:b00c:0:25de
CNAME	pss.alicdn.com	pss.alicdn.com.gds.alibabadns.com
MX	google.com	smtp.google.com
NS	google.com	ns1.google.com ns2.google.com ns3.google.com ns4.google.com
TXT	google.com	google-site-verification=TV9-DBe4R80X4v0M4U_bd_J9cpOJM0nikft0jAgjmsQ webexdomainverification.8YX6G=6e6922db-e3e6-4a36-904e-a805c28087fa v=spf1 include:_spf.google.com all MS=E4A68B9AB2BB9670BCE15412F62916164C0B20BB docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c1l2BPvqKX8= facebook-domain-verification=22rm551cu4k0ab0bxsw536tlds4h95

improve the ability of our encoder. Since most DNS tunnels encode query data into the subdomain portion of the fully qualified domain name (FQDN), we remove the query content of the top-level domain name (TLD) and the second-level domain name (2LD) (lines 2–5 of Algorithm 1), and only reserve the subdomain part. In addition, we removed those noisy characters such as '.', ':' and formatted all letters to lowercase (lines 10–15 of Algorithm 1). Since the multi-head attention mechanism encoder can only accept input from stream data, we design our word-to-index mapping to convert the content of the Q&R pair into representational vectors (line 17). The keys of the mapping are composed of numbers, lowercase letters, and underscores. We use the 'Tab' for query and response content splitting, and their corresponding values are numbers of range [0,38].

In a legitimate DNS channel, DNS payloads usually contain human-readable strings. However, the DNS tunnel payloads are strings of random characters and are not human-readable. In our system, we build a recurrent neural network (RNN) encoder with a multi-head attention mechanism to distinguish between the content of the Q&R pair in DNS tunnels and the content of the Q&R pair in normal DNS traffic. Recurrent neural network (RNN) models can remember the previous state and apply it to the output calculation, and have been widely adopted for sequence data encoding. Using the attention mechanism can effectively improve the shortcomings of slow training of the RNN model and enhance the learning ability of the RNN model (Vaswani et al., 2017). It is considered an effective method to solve some sequence content detection problems. Recently, the attention mechanism has been widely used in the domain of cyber security (Kozik et al., 2021; Ranade et al., 2021) and network traffic detection (He et al., 2020) and has been proved to be effective.

Fig. 4 shows the architecture of the multi-head attention mechanism encoder. We use each pre-processed content vector $X_i = [x_1, x_2, \ldots, x_n]$ as a single raw input to the multi-head attention encoder. Since we adopt the attention mechanism of parallel computing, we need to add position encoding information to each word encoding in the content vector in advance. We use the four-dimensional position encoding information, and obtain a new position encoding word vector x_i by accumulating with each word encoding.

Then in the multi-head attention mechanism, we first initialize 8 groups of attention heads containing query vector W^Q, key vector W^K, and value vector W^V. Each embedding vector x_i is projected to these vectors using linear transformations, i.e., $Q_i = W^Q x_i$, $K_i = W^K x_i$, $V_i = W^V x_i$. The purpose of linear projections is to generate the inputs for the self-attention mechanism, which is to figure the compatibility between each input x_i and all the other inputs $x_i \sim x_k$ via a similarity function shown in formula (1).

$$att_i = \sum_{j=1}^{k} \frac{\frac{Q_i K_j^T}{\sqrt{d_k}}}{z} \cdot V_j \tag{1}$$

Table 3
statistical traffic features selected in our system .

Feature Groups	Feature Name
Single packet	Length of query payload
	Length of the response payload
	Average size of domain name TTL
Interaction	Ratio of answers to queries
	Time interval of the query to response
	Ratio of upload to download size

K_j is a d_k dimensional vector, and z is the normalization factor. It should be noticed that not every input vector is needed for the self-attention calculation. An optional masking strategy that randomly ignores a few inputs while generating attention vectors is allowed to avoid over-fitting. Afterward, for the 8 sets of attention heads, the conjunction operator $att_i = att_{i,1} \bigoplus att_{i,2} \bigoplus \ldots \bigoplus att_{i,8}$ is used to obtain the attention vectors. The ReLU activation function is used in the feed-forward network.

$$h_i = max(0, W_1 att_i + b_1) + b_2 \tag{2}$$

W_1, b_1, and b_2 are the adjustable hyperparameters, and each content vector is transformed into a two-dimensional coded representation vector through the layer normalization.

3.3. Statistical traffic features extraction

In our work, we divide the statistical traffic features into two categories, single packet features and interaction features, according to the features present in a single packet and the features present in the query-response dialogue. The features we extracted are listed in Table 3.

Length of the query payload. The Query payload is the payload part of the DNS query packet after removing the header. During the DNS tunnelling, the controlled end wraps the transmitted content in the query packet. To carry as much information as possible during a single transmission, the query packet in DNS tunnelling is often larger than that of the normal channel. Therefore, the length of the query payload acts as a good indicator of DNS tunnelling.
Length of the response payload. The response payload is the payload part of the DNS response packet after removing the header. In DNS tunnelling, the attacker usually sends two kinds of data, one is the short heartbeat data to determine whether the connection exists, and the other is the encoded long attack command. On the other hand, for normal DNS traffic, the data returned by the DNS server mostly has 4 bytes (in IPv4 format) or 16 bytes (in IPv6 format). Thus the normal DNS response packet size is between the size of the short heartbeat data size and the attack command size of the tunnelling response packet. Therefore, the length of the response payload is capable of distinguishing normal packets and DNS tunnelling packets. **The average size of domain name TTL.**

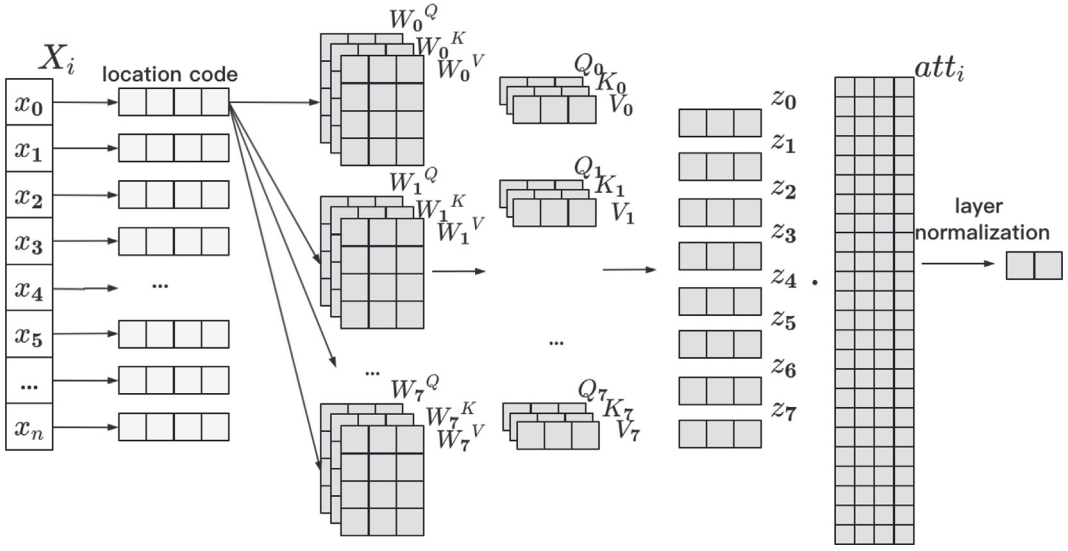

Fig. 4. Overview of the multi-head attention mechanism encoder.

Domain name time to live (TTL) is the DNS record caching time on the DNS server. Setting a higher value for the domain name TTL is good for improving the speed of DNS queries and can against DDOS attacks to some extent. However, the way DNS tunnelling communicate determines that the sub-domains used for communication are different each time, and thus each subdomain is usually assigned a smaller TTL value. Therefore, the size of the TTL can distinguish between normal packets and DNS tunnelling packets. **The ratio of answers to queries.** One IP may correspond to multiple canonical names (CNAME), and a single query may receive more than one answer resource record. However, DNS tunnelling data often includes only one answer record without authorization resource record and an additional resource record. Therefore, the ratio of answers to queries could help distinguish normal packets from DNS tunnelling packets. **The time interval between query and response.** To increase the stability of the tunnelling, many tunnelling tools will force users to use the public IP to indicate the address of the attacker's domain name server. In this way, DNS tunnelling will simplify the process of recursive resolution of their domain names. In a normal DNS channel, when the query is not hit by the local cache, the time interval between the query and the response packet is usually longer due to the need to recursively resolve the subdomain name. It is thus indicative to use the time interval between a query and its corresponding response as a feature. **The ratio of upload to download size.** When in the DNS tunnelling, the attacker server sends a small packet with control commands to the controlled client-side, which needs to send back a big packet with sensitive resource file data. However, the opposite is true for normal DNS resolution, where the client query message is usually short and the DNS server returns more data information. As a result, DNS tunnelling sessions have a larger proportion of upload to download size than normal DNS sessions.

3.4. XGBoost Classifier

In DNS tunnel detection, due to the large amount of low complexity training data, the trained model is easy to get overfitting. At the same time, in order to meet the industry's requirements for the effectiveness of DNS tunnelling detection, the detection cost needs to be as low as possible. Considering that the eXtreme Gradient Boosting (XGBoost) (Chen and Guestrin, 2016) algorithm is more robust than traditional machine learning algorithms, and its engineering optimization can reduce computing costs, we chooses the XGBoost algorithm to build the classification model.

In our work, the dataset is represented in the format shown in formula (3), where $x_t = [x_0, x_1, \ldots, x_8]$ is a fusion vector composed of a six-dimensional statistical feature vector and a two-dimensional encoding feature vector, $y_t \in \{0, 1\}$ is a scalar with the output of integer-value as its element. And n is the number of samples in the training dataset.

$$D = (x_t, y_t), t = 1, 2, \ldots, n \tag{3}$$

The objective function is shown in Eq. 4.

$$Obj = L + \Omega \tag{4}$$

The objective function is composed of two parts, i.e., the differential convex loss function $L = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i\right)$, which measures the difference between the predicted value $\hat{y}_i$ and the actual value y_i, and the regularization penalty function $\Omega = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$, which is used to prevent the model from over-fitting. T is the number of leaves in the tree, w is the weight of the leaf nodes, γ and λ are configurable parameters, and their accumulation are used to evaluate the complexity of the tree.

Each iteration of XGBoost will traverse the eigenvalues $x_{t,z} (t = 1, 2, ., n, z = 1, 2, \ldots, 8)$ of each feature, and the greedy search can minimize the split point of the objective function value, and thus complete the splitting of leaf nodes. We used a total of 150 decision trees to integrate our XGBoost classifier, and set the maximum depth of each tree to 4. Therefore, when the depth of the decision tree reaches 4, it stops splitting and generates a new decision tree. When the number of trees reaches 150, it stops generating trees, and all the trees add up to form our XGBoost classifier. We use the classification results of all boosts as the final classification result. When the classification result approaches 1, the input sample is regarded malicious, and when it is 0, the input sample is considered a legal sample. We use cross-validation to search for hyperparameters on the XGBoost classifier, and organize the parameter values in Table 4.

4. Evaluation

Our experiment with the system is designed to answer the following three research questions.

RQ1: How effective is our approach compared with state-of-the-art approaches?

Table 4
Hyperparameter search .

Hyperparameters	Search Range	Parameter Value
max_depth	[3, 10]	4
n_estimators	[100, 1000]	150
min_child_weight	[1, 10]	6
subsample	[0.5, 1.0]	1.0
colsample_bytree	[0.5, 1.0]	1.0
gamma	[0, 5]	0
reg_alpha	[0, 1)	0
reg_lambda	[0, 1)	1
learning_rate	[0.001, 0.1]	0.03

RQ2: How each component of our approach contributes the detection results.

RQ3: How effective is our approach in real-world DNS tunneling detection?

4.1. Experiment settings

4.1.1. Experiment environment

We run our offline experiment on a server with 2 Intel Xeon Platinum 8260 CPU @2.30GHz, 8 NVIDIA GeForce RTX2080 Ti GPU (each graphics card memory is 11GB) and 512GB RAM. For the real-world online test environment, a single network node deployed by a Party B security vendor is used, and its network environment is 314Mbps (peak value). We implement our method with Python version 3.8.6 and Pytorch version 1.8.0.

4.1.2. Compared method

We compare with three state-of-the-art approaches fixed grouped packets detection method and single-packet detection methods, i.e., the LSTM-based approach (Chen et al., 2021), the feature-level CNN (D'Angelo et al., 2022), and the grouped packets detection method based on Random Forest (Yang et al., 2020). Since those approaches do not release their source code, we implement the models in each approach strictly follow the descriptions in the corresponding paper. The LSTM model contains an embedding layer and a hidden layer, and the Adam Optimizer is used as the optimizer. The feature-level CNN model contains 2 convolutional layers, 2 max-pooling layers, 1 softmax layer, and uses Adagrad Optimizer as the optimizer. The Random Forest model contains 6 max features, 10 estimators and 150 max depth. Note that, we conduct experiments following the optimal parameters and configurations reported in the relevant method papers.

4.1.3. Dataset

Our offline datasets consist of a base dataset and a generalization test dataset, and the metadata for the datasets are shown in Table 5.

Base dataset. We adopt the open-source DNS tunnelling traffic (Chen et al., 2021), which was collected using 8 different DNS tunnelling tools in a simulated environment, as part of the malicious samples in our base dataset. To get real benign DNS traffic as much as possible, we use a transparent certificate query to obtain 100K sub-sites of Alexa top 1K websites and collect DNS data packets by visiting them in the campus environment. Since the number of packets collected by Alexa is unbalanced with the open malicious packets, we also use manual screening to extract DNS packets from daily traffic to expand our benign sample set.

Generalized test dataset. We find that most of the tunnelling packets in the base malicious dataset are heartbeat packets, which are used to maintain connectivity and without actual attack payload. To better simulate the real attack, we collect 150 commands commonly used by hackers for intranet penetration on Linux and Windows systems and use these commands in different DNS tunnelling tools to obtain malicious samples for our more challenging generalization test. In addition, the base benign sample dataset is harvested from a highly restrictive campus environment that is hardly representative of an environment that requires a confrontation with DNS tunnelling. To test the robustness of the model in more complex environments, we obtain manually screened DNS packets from a security company as our benign samples for generalization testing.

4.1.4. Data pre-processing

Since the collected offline DNS data is saved in the form of pcap files, the input of most DNS tunnelling detection models is not the original pcap file, we have performed different data prep-rocessing for different compared approaches. We used "Tshark" to parse our raw pcap files. Since the input to the bytes-level CNN model (Liu et al., 2019) is a binary sequence, we split out individual DNS packets from the pcap file and extract the first 300 bytes of each DNS packet. For the feature-level CNN model (D'Angelo et al., 2022), we extracted 22 different features from DNS packets and transformed them into a 6x-dimensional image matrix. While the LSTM model (Chen et al., 2021) uses the FQDN in the DNS query packet as input, we filter out all DNS query packets from the pcap file and extract the query content from the payload of each DNS query packet. For our system, we rely on the extended information to extract all DNS Q&R pairs from the existing pcap packets and sequentially extract the corresponding detection content.

4.1.5. Metrics

We use the matrices of Accuracy(A), Precision (P), Recall (R), and F1-Score (F) to evaluate the effect of the model in the offline dataset. The formulas used to calculate the metrics are shown in Formulas (5) ∼ Formulas (8), where TP represents the number of samples correctly classified as positive, FN represents the number of samples incorrectly classified as negative, FP represents the number of samples incorrectly classified as positive, and TN represents the number of samples that are the correctly classified negative.

$$A = \frac{TP + TN}{TP + FP + FN + TN} \tag{5}$$

$$P = \frac{TP}{TP + FP} \tag{6}$$

$$R = \frac{TP}{TP + FN} \tag{7}$$

$$F = \frac{2 * P * R}{P + R} \tag{8}$$

4.2. Experimental results

RQ1: Comparison with state-of-the-art approaches.

External experiments compare the performance of our system with three other systems that perform well in grouped packets detection methods and single packet detection methods. In particular, we compare our system with the bytes-level CNN model Liu et al. (2019), features-level CNN model D'Angelo et al. (2022), and the LSTM model Chen et al. (2021) on the base and generalization test datasets.

The experimental results of each detection method are shown in Table 6. We can observe that our method achieves the best results on all metrics for both the base test set and the generalized

Table 5
The metadata of the offline datasets.

Datasets	Category	Composition (Packets)	Experimental division (Q&Rs)
Base dataset	malicious	Open-set (600K)	Train:180K Validation:20K Test:100K
	benign	Alexa (200K), Campus (400K)	Train:180K Validation:20K Test:100K
Generalization test dataset	malicious	Simulated attack (200K)	Total:100K
	benign	Security company (200K)	Total:100K

Table 6
Comparison results with state-of-the-art detection method.

Datasets	Metrics	Ours	LSTM	Feature-Level CNN	Random Forest
Base dataset	*A*	99.68%	95.45%	97.36%	98.25%
	P	99.57%	96.14%	98.24%	98.16%
	R	99.79%	94.77%	96.48%	98.33%
	F	99.68%	95.45%	97.35%	98.24%
Generalized test dataset	*A*	98.82%	73.65%	88.81%	81.27%
	P	98.77%	74.55%	89.48%	87.23%
	R	98.87%	71.31%	88.14%	75.32%
	F	98.82%	72.90%	88.80%	80.84%

Table 7
Average execution time (seconds).

	LSTM	Feature-Level CNN	Random Forest	Ours
traffic grouping(s)	-	-	103.40	32.00
features calculation(s)	**0.21**	0.32	0.14	1.41
online detection(s)	1.87	2.22	0.002	**0.001**

Table 8
The ablation study results for different features.

	Encoding features	Statistical features	Combined features
A_{Base}	94.32%	93.54%	**99.68%**
A_{Gen}	84.13%	79.54%	**98.82%**

Table 9
Effect comparisons of different packets in DNS.

	query packet	response packet	Q&R pairs
A_{Base}	96.32%	97.12%	**99.68%**
A_{Gen}	75.21%	80.75%	**98.82%**

test dataset, and the advantages are more obvious on the generalized dataset. The LSTM model, feature-level CNN model and Random Forest achieves good accuracy and F1-score on the base test dataset, yet there is a large decline (22.55%, 8.55% and 17.4% on F1-score) on the generalized test dataset. This shows that those methods suffer from poor generalizability. Our approach remains good results on all four metrics on the generalized test dataset, which shows good generalizability.

Except for detection effectiveness, we also measure the efficiency of all compared method. In particular, we report the time used to process 10k packets for each method. The results are shown in Table 7. Compared with the single packet detection method, our method consumes less online time (including feature calculation and online detection). Compared with the grouped packets detection method, our method consumes more online time, but the time consumed for traffic grouping is reduced by 69%. In terms of the total detection time, our method reduces by 66.77%.

RQ2: The ablation study results.

We conduct an ablation study to answer the second research question. In particular, we measure how the two-dimensional encoding feature and the six-dimensional statistical feature individually perform, as well as how do they perform combined together, on the offline task. We use the baseline dataset for offline task evaluation. In particular, we divide the base data set into the training set, validation set, and test set according to the ratio of 8:1:1 and performed standard 10-fold cross-validation to train the decision model.

The experimental results of the ablation study are shown in Table 8. A_{Base} indicates the accuracy of the different methods on the base test dataset, and A_{Gen} indicates their accuracy on the generalization test dataset. Note that we use the training set of the base dataset to train the detection model, and report the accuracy on the test set of the base dataset (A_{Base}) as well as the accuracy on the generalization test dataset (A_{Gen}). We can observe that, using only encoding features or statistical features achieve lower accuracy than using combined features on both the base dataset and generalization test dataset. The accuracy reduced on the generalization test dataset is 10.19%, 14% and 0.84% using only encoding features, statistical features, and combined features, respectively. The results show that using combined features can achieve the best detection accuracy, as well as the best generalizability.

We also evaluate the effectiveness of Q&R pairs in anomaly detection. In particular, we compare the anomaly detection accuracy with only DNS query packets, DNS response packets, and Q&R pair as the basic unit for feature extraction. The encoding features for the single packet are extracted exactly the same as that of Q&R pair. However, due to the lack of session behavior in single packet, we only extract the query content payload length for the query packet. For the response packet, only the length of the response content and the average TTL value of the domain name are extracted.

Table 9 shows the evaluation results. We can observe that using Q&R pair achieves the highest accuracy in both the base dataset and the generalization dataset. Using either the query packet or the response packet as the detection unit results in an accuracy reduction of more than 3% and 2%, respectively. Moreover, using Q&R pair shows good generalization ability, while the other two detection units show dramatically large decrease on detection accuracy (more than 21% and 16%, respectively). The evaluation results show the effectiveness of using Q&R pair as the basic detection unit.

Table 10
Effect comparisons of encoding feature extraction methods.

	Information volume analysis	Character statistics	attention-mechanism encoder
A_{Base}	97.61%	97.56%	**99.68%**
A_{Gen}	82.13%	85.62%	**98.82%**

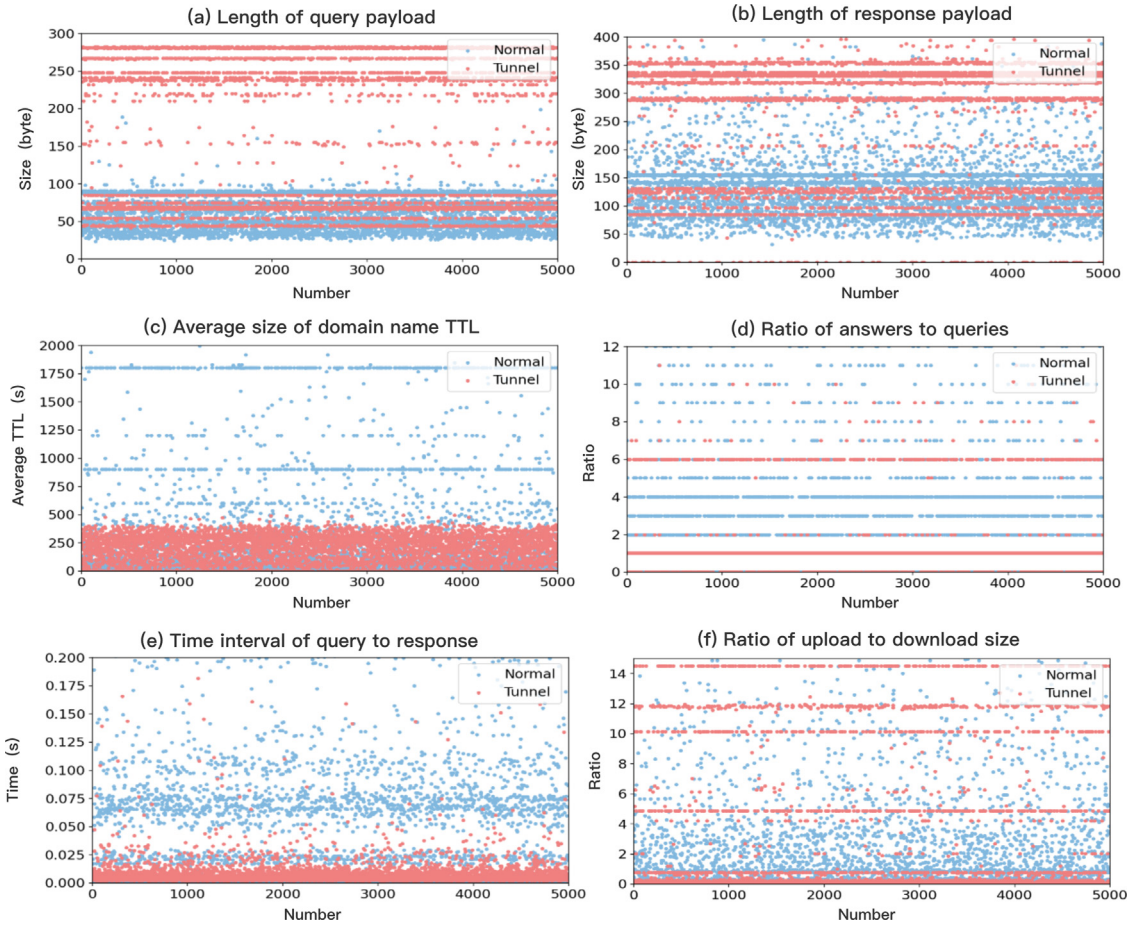

Fig. 5. Plot visualization of each statistical feature.

We further evaluate the our encoding feature extraction method with standard encoding feature extraction method to see the performance difference, and the results are reported in Table 10. For statistical features, we randomly sample the data from normal traffic and tunnelling traffic, and visualize each individual feature. Fig. 5 shows the visualization results.

In Table 10, from the experimental results of the three encoding feature extraction methods, the methods using information volume analysis and character statistics have similar performance on the base and generalization test dataset, they both get a good performance on the base test dataset, but only 82.13% and 85.62% accuracy on the generalization test dataset. Our method of encoding feature extraction using the attention-mechanism encoder maintains accuracy of up to 99.68% and 98.82% on both the base and generalization datasets. Therefore, the use of an attention-mechanism encoder for encoding feature extraction is effective and has better generalizability.

Fig. 5 shows the data plots of each statistical feature by randomly sample 5k data from all samples in the base dataset and the generalization test dataset. The results show that the DNS tunnelling traffic and the normal DNS traffic have a good degree of discrimination on the average size of domain name TTL and the time interval of a query to response. For the ratio of answers to queries and the ratio of upload to download size, the normal and the tunnelling data are relatively well separated. For the length of query payload and the length of response payload, there are around 15% and 17% data points overlapping between the normal and tunnelling data. Yet the two statistical features can still provide some information combined with other features, and thus we keep those two features.

In XGBoost, we consider the feature to guide the tree node splitting as effective. In the experiment, we use the feature frequency to represent the relative importance of a specific feature in XGBoost. The frequency is calculated as the number of splits by a particular feature to the total number of splits. Fig. 6 shows the frequency of each feature in the XGBoost classifier. We can observe that the feature frequency of encoding features in the XGBoost classifier is 53.33%, and the feature frequency of statistical features is 46.67%. Among the statistical features, the frequency of query payload length, response payload length, and domain name average TTL are 11.11%, 13.33% and 13.33%, respectively. The feature frequencies of the other three statistical features are 2.22%, 2.22%, and 4.44%. The results show that both encoding features and statistical features contributes to the decision point of the XGBoost classifier.

RQ3: The experiment on the real-world application.

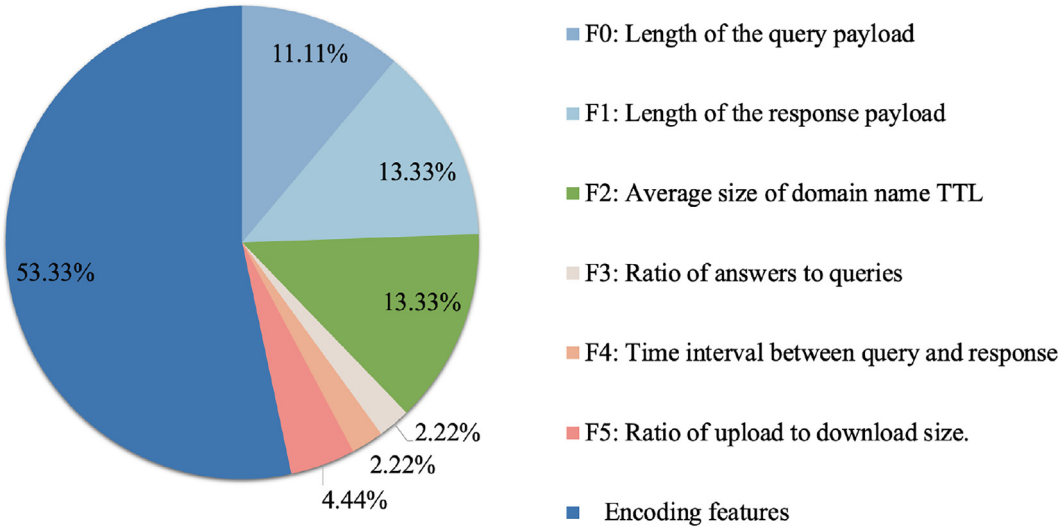

Fig. 6. Frequency of each feature in the XGBoost classifier.

Table 11
DNS tunnelling in a real-world network.

Type	Query content	Answer content
Trojan.Win32.lsmdoor.gen	n.n.c.F453136668BA4021BC9D063179D E678E.dnslookupdater.com TTpDQz8!.0.dr.F453136668BA4021BC9D	a67d:db8:a2a1:7334:7654:4325:370:2aa3
063179DE678E.dnslookupdater.com	a67d:db8:85a3:4325:7654:8a2a:370:7334 n.1.f.F453136668BA4021BC9D063179D	
E678E.dnslookupdater.com	2020:2020:2020:2020:2020:2020:2020:2020	
Helminth	00500000LFI232A616131.go0gle.com	119.104.111.97
	00500000LFI232A616132.go0gle.com	109.105.32.32

We test our system in a campus environment protected by a full-flow security analysis platform. During the one month actual traffic detection period, our system found 1287 suspicious DNS tunnelling traffic. We sent these suspicious traffic to security experts, 14 of which are considered to be normal traffic generated false positives, while the remaining 1273 traffics are confirmed to belong to 7 DNS tunnelling events, we show some DNS tunnelling examples in the table 11.

After an expert system review, we have found that the traffic belonging to DNS tunnelling traffic is generated by the malware "Trojan.Win32.Ismdoor.gen" and the DNS tunnelling tool "Helminth", which is commonly used by the APT group "OilRig". The answer content "119.104.111.97" and "109.105.32.32" in the "Helminth" DNS traffic will become the "whoami" command after being decoded by Base10 to obtain computer-related information in turn. The "TTpDQz8!" in the "Trojan.Win32.Ismdoor.gen" DNS traffic is formed by the handshake query "M:CC?" of the C&C host through Base64 encoding.

5. Related work

We summarize related work in Table 12. In particular, we explicitly list the basic detection unit, the algorithms/models, the malicious dataset as well as the weakness of each approach.

5.1. Single-packet-based detection

Single packet based detection methods determine whether the content of a single DNS packet is encoded. The common detection content is the payload of DNS packets, such as the queries or RRs.

These methods use queries as the detection content, assuming that the data sent by the client is usually only encoded in the subdomain of the FQDN. Qi et al. (2013) believed that the distribution of characters in the normal subdomain generally conforms to the distribution of natural language, while the distribution of characters in the DNS tunnel subdomain tends to be more random. They designed a scoring mechanism based on the frequency of bigram characters. However single frequency is not a good characterization of the encoding behaviour at some point. Ahmed et al. (2019) extracted 8-dimensional features of the FQDN and constructed a dataset of benign FQDNs to train an isolation forest model to detect the DNS tunnel FQDNs. With the development of deep learning, there are more and more attempts to detect DNS tunnel FQDNs without relying on feature engineering. Zhang et al. (2019) used various deep learning models, such as DNN, CNN, and RNN models, to check whether the FQDN of a DNS packet belongs to a normal DNS traffic or a DNS tunnel. Chen et al. (2021) used the LSTM model with a whitelist Group Policy to reduce the false alarm rate of the system.

Those methods use RRs as the detection content, mainly detecting DNS tunnels by the RRs of the DNS response packet. Das et al. (2017) focuses on RRs of TXT type, 10-dimensional features are extracted from each RR, and the detection of RRs belonging to DNS tunnelling is performed by a linear regularized logistic regression model. To make the detection not only limited to certain specific types of corresponding, Liu et al. (2019) used the one-hot method to encode the byte sequences of DNS packets and used the CNN model to judge whether the packets were from DNS tunnelling. And D'Angelo et al. (2022) extracted a featured image containing 22-dimensional features for each response packet and used a CNN model to determine whether the packets were from DNS tunnelling.

Single packet based methods do not require the acquisition of associated additional packets and allow real-time monitoring of the DNS tunnelling. However, lacking the traffic characteristics of the DNS tunnelling, they tend to have a high false alarm rate.

Table 12
Summary of related work.

Detection unit	Paper	Algorithms	Dataset(malicious)	Weakness
Single packet	Qi et al. (2013)	The expect value of bigram character frequency	Dnscat	Poor performance with long DNS tunnel domains
	Ahmed et al. (2019)	Isolation Forest	DET	High false positive rate
	Zhang et al. (2019)	Dense Neural Network, Recurrent Neural Network, one-dimensional CNN	Iodine, Dns2tcp, DNS reverse shell, Dnscat2	1. Lack of interpretability 2. High requirements of the hardware
	Chen et al. (2021)	Long short-term memory	Iodine, Dnscat2, Ozymandns, Dns2tcp, Cobaltstrike, DNShell, DET, DNSExfiltrator	Depend on non-real-time components
	Das et al. (2017)	Logistic Regression, K-means	Dnscat, Bernhardpos	1. Vunerable to different evasion technology 2. Detection of TXT records tunnelling is under-performance
	Liu et al. (2019)	Byte-level Convolutional, Neural Networks	Iodine,Dns2tcp, ReverseDNShell, OzymanDNS, Dnscat2	Vunerable to DNS tunneling short packets
	D'Angelo et al. (2022)	Feature-image-level, Convolutional, Neural Networks	Iodine, Dnscat, Ozyman	Single packet high false positive
Grouped packets	Aiello et al. (2016, 2014, 2015)	Bayes, KNN, Neural Networks, SVM	Dns2tcp	Poor generality of the model
	Binsalleeh et al. (2014)	Custom Pattern Recognition	Malware samples	poor applicability
	Yang et al. (2020)	Random forest	-	Poor real-time performance

And in theory, it is also possible for attackers to defeat detection through domain name spoofing(Wang et al., 2021).

5.2. Grouped-packets-based detection

Grouped packets based detection methods usually group the DNS packets into sets and extract features on each set. The grouping conditions mainly include the data slices, domains of FQDNs, IPs, etc.

Aiello et al. (2016, 2014, 2015) tried to identify possible DNS tunnels from the DNS traffic segments when the server was working. They used the packet sets composed of n pairs of packets as determination units, started with a two-stage classifier, and have since expanded their feature set and optimized the classifier. However, it is difficult to obtain more features across different datasets as the data in the collection may not be very correlated. To allow for better correlation of data between groups, Binsalleeh et al. (2014) used the FQDN as the group unit, and make determinations by the different resource records. To characterize the session information of DNS tunnelling, some researchers further used not only domain names but also IPs as the basis of grouping. Yang et al. (2020) grouped the FQDNs of DNS packets based on the same request IP and the same domain name. Then concatenated the subdomain of the FQDNs into a string and extracted 9-dimensional features from the concatenated string. It could group the traffic based on the client and the server of a DNS tunnelling at the same time.

Grouped packets based detection often captures additional statistical features of the DNS traffic. While more features allow them to achieve higher detection rates, the detection efficiency is often low, and it is difficult to meet the requirements of enterprises to implement detection.

6. Conclusion

In this paper, we propose a method to identify DNS tunnelling traffic based on encoding and statistical features extracted from Q&R pairs. With DNS Q&R pairs, we can extract encoding and behavioral statistical feature segments in small detection units, which achieves good detection efficiency while improving the detection effectiveness. We use the Attention encoder to extract the encoding features of DNS Q&R pairs, and manually analyze and extract prior statistical features. XGBoost is adopted to classify DNS tunnelling traffic and legitimate DNS traffic based on the above features. We use the open-source DNS tunnelling traffic and the subdomain of Alexa top website to build our basic dataset. We build our generalization test dataset by mixing simulated attack traffic collected using DNS tunnelling tools with daily DNS traffic from a security company. Experiments show that our system has an accuracy rate of 99.68% on the basic data set, and can maintain an accuracy rate of 98.82% on generalization tests. In the efficiency comparison experiment with the package detection method, we reduced the time consumption of the detection process by 66.77%. In addition, our system also maintains good robustness in real-world traffic from a campus environment and finds 7 DNS tunnelling events.

There are still points that can be improved and further discussed in our method. First, we notice that although the Attention encoder have good accuracy among current encoding feature extraction methods, the computational cost of Attention encoder is relatively high, resulting in long online computation process. Therefore, continuing to study the encoding feature extraction method in the DNS tunnel and improving the detection efficiency of our method is direction that we need to further explore. Currently, we only extract six detection dimensions from the behavioral statistical features of DNS tunnelling. Although it is proven to work most of the time, it is still possible of being confused by attackers. Therefore, at the level of behavioral statistics, extracting more detection dimensions is subject to our future work.

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.

CRediT authorship contribution statement

Yu Tu: Writing – original draft, Validation, Formal analysis, Visualization, Software, Methodology, Investigation, Conceptualization, Data curation. **Shuang Liu:** Methodology, Writing – review

& editing, Validation, Funding acquisition, Resources, Supervision, Project administration. **Qian Sun:** Resources, Data curation, Visualization, Investigation.

Data Availability

Data will be made available on request.

Acknowledgement

This work was supported by National Natural Science Foundation of China (grant nos. U1836214), Key Laboratory of Safety-Critical Software (Nanjing University of Aeronautics and Astronautics), Ministry of Industry and Information Technology Open Project no. NJ2022027.

References

Ahmed, J., Gharakheili, H.H., Raza, Q., Russell, C., Sivaraman, V., 2019. Real-time detection of dns exfiltration and tunneling from enterprise networks. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, pp. 649–653.

Aiello, M., Mongelli, M., Cambiaso, E., Papaleo, G., 2016. Profiling dns tunneling attacks with pca and mutual information. Log. J. IGPL 24 (6), 957–970.

Aiello, M., Mongelli, M., Papaleo, G., 2014. Supervised learning approaches with majority voting for dns tunneling detection. In: International Joint Conference SOCO14-CISIS14-ICEUTE14. Springer, pp. 463–472.

Aiello, M., Mongelli, M., Papaleo, G., 2015. Dns tunneling detection through statistical fingerprints of protocol messages and machine learning. Int. J. Commun. Syst. 28 (14), 1987–2002.

Bienaimé, P., Mazon, P., 2011. Dns tunneling with scapy. https://code.google.com/archive/p/dnscapy/.

Binsalleeh, H., Kara, A.M., Youssef, A., Debbabi, M., 2014. Characterization of covert channels in dns. In: 2014 6th International Conference on New Technologies, Mobility and Security (NTMS). IEEE, pp. 1–5.

Centre, N. C. S., 2020. Advisory: Apt29 targets covid-19 vaccine development. https://www.ncsc.gov.uk/news/advisory-apt29-targets-covid-19-vaccine-development.

Chen, S., Lang, B., Liu, H., Li, D., Gao, C., 2021. Dns covert channel detection method using the lstm model. Comput. Secur. 104, 102095.

Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785–794.

Cian Lynch, D. A., Teodorescu, C., 2016. Multigrain point of sale attackers make an unhealthy addition to the pantry. https://www.fireeye.com/blog/threat-research/2016/04/multigrain_pointo.html.

D'Angelo, G., Castiglione, A., Palmieri, F., 2022. Dns tunnels detection via dns-images. Informa. Process. Manag. 59 (3), 102930.

Das, A., Shen, M.-Y., Shashanka, M., Wang, J., 2017. Detection of exfiltration and tunneling over dns. In: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, pp. 737–742.

Ekman, E., 2021. Iodine. https://github.com/yarrick/iodine.

Group, N. W., 1987. Domain names - concepts and facilities. https://datatracker.ietf.org/doc/html/rfc1034.

Grunzweig, J., 2016. New wekby attacks use dns requests as command and control mechanism. https://unit42.paloaltonetworks.com/unit42-new-wekby-attacks-use-dns-requests-as-command-and-control-mechanism.

He, H.Y., Yang, Z.G., Chen, X.N., 2020. Pert: Payload encoding representation from transformer for encrypted traffic classification. In: 2020 ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K). IEEE, pp. 1–8.

Infoblox, 2016. Infoblox security assessment report. https://www.infoblox.com/wp-content/uploads/infoblox-security-assessment-report-2016q2.pdf.

Ishikura, N., Kondo, D., Vassiliades, V., Iordanov, I., Tode, H., 2021. Dns tunneling detection by cache-property-aware features. IEEE Trans. Netw. Serv. Manage. 18 (2), 1203–1217.

janprunk, 2021. Ozymandns. https://github.com/janprunk/ozymandns.

Jay Carlson, Ryan O'Horo, P., Stalmans, E., 2017. Det (extensible) data exfiltration toolkit. https://github.com/sensepost/DET.

Kaspersky, 2017. The epic turla (snake/uroburos) attacks. https://www.kaspersky.com/resource-center/threats/epic-turla-snake-malware-attacks.

Kozik, R., Pawlicki, M., Choraś, M., 2021. A new method of hybrid time window embedding with transformer-based traffic data classification in iot-networked environment. Pattern Analysis and Applications 1–9.

Liu, C., Dai, L., Cui, W., Lin, T., 2019. A byte-level cnn method to detect dns tunnels. In: 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC). IEEE, pp. 1–8.

Merlo, A., Papaleo, G., Veneziano, S., Aiello, M., 2011. A Comparative Performance Evaluation of Dns Tunneling Tools. In: Computational Intelligence in Security for Information Systems. Springer, pp. 84–91.

Qi, C., Chen, X., Xu, C., Shi, J., Liu, P., 2013. A bigram based real time dns tunnel detection approach. Procedia Comput Sci 17, 852–860.

Raman, D., De Sutter, B., Coppens, B., Volckaert, S., De Bosschere, K., Danhieux, P., Van Buggenhout, E., 2012. Dns tunneling for network penetration. In: International Conference on Information Security and Cryptology. Springer, pp. 65–77.

Ranade, P., Piplai, A., Mittal, S., Joshi, A., Finin, T., 2021. Generating fake cyber threat intelligence using transformer-based models. arXiv preprint arXiv:2102.04351.

Ron, 2021. Dnscat2. https://github.com/iagox86/dnscat2.

Saeli, S., Bisio, F., Lombardo, P., Massa, D., 2020. Dns covert channel detection via behavioral analysis: a machine learning approach. arXiv preprint arXiv:2010.01582.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. In: Advances in neural information processing systems, pp. 5998–6008.

Wang, Y., Zhou, A., Liao, S., Zheng, R., Hu, R., Zhang, L., 2021. A comprehensive survey on dns tunnel detection. Comput. Netw. 108322.

Yang, Z., Hongzhi, Y., Lingzi, L., Cheng, H., Tao, Z., 2020. Detecting dns tunnels using session behavior and random forest method. In: 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC). IEEE, pp. 45–52.

Yunakovsky, S., Pomerantsev, I., 2017. Use of dns tunneling for c&c communications. https://securelist.com/use-of-dns-tunneling-for-cc-communications/78203/.

YUNAKOVSKY, S., POMERANTSEV, I., 2018. Denis and co.https://securelist.com/denis-and-company/83671/.

Zander, S., Armitage, G., Branch, P., 2007. A survey of covert channels and countermeasures in computer network protocols. IEEE Commun. Surv. Tutor. 9 (3), 44–57. doi:10.1109/COMST.2007.4317620.

Zhang, J., Yang, L., Yu, S., Ma, J., 2019. A dns tunneling detection method based on deep learning models to prevent data exfiltration. In: International Conference on Network and System Security. Springer, pp. 520–535.

Yu Tu received the bachelors degree from the School of Automation, Wuhan University of Technology in 2020.And he is now a master student of the College of Intelligence and Computing, Tianjin University. His research interests include network intrusion detection and anomalous traffic detection.

Shuang Liu is an associate professor at Tianjin University (TJU), China. She received PhD degree from National University of Singapore in 2015. She worked as a research fellow in SUTD during 2015–2016, and lecturer in SiT during 2016–2017. She has been a faculty member of TJU since 2018. Her research interests focus on software quality assurance and security in general.

Qian Sun received her PhD degree in computer science from Nanyang Technological University, Singapore, 2014. She worked as a research fellow in Fraunhofer Singapore during 2014–2017, and an associate professor in Tianjin University during 2017–2021. She is now an associate professor at the School of Electronic and Information Engineering, Nanjing University of Information Science and Technology. Her current research interests include intelligent graphics, deep learning, and adversarial example recognition.