ITransformer_CNN: A Malicious DNS Detection Method with Flexible Feature Extraction

Wenyue Zhang  
Nanjing University of Posts and Telecommunications

Wei Chen  
(✉ chenwei@njupt.edu.cn)  
Nanjing University of Posts and Telecommunications

Zhen Zhang  
Nanjing University of Posts and Telecommunications

Fan Zhang  
Wuhan Polytechnic University

Lifa Wu  
Nanjing University of Posts and Telecommunications

Research Article

Keywords: DNS security, feature extraction, Transformer, multi-head attention, convolutional neural network

Posted Date: October 5th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2103771/v1

License: ©️ This work is licensed under a Creative Commons Attribution 4.0 International License. 
Read Full License
ITransformer_CNN: A Malicious DNS Detection Method with Flexible Feature Extraction

Wenyue Zhang\textsuperscript{1}, Wei Chen\textsuperscript{1*}, Zhen Zhang\textsuperscript{1}, Fan Zhang\textsuperscript{2} and Lifa Wu\textsuperscript{1}

\textsuperscript{1}School of Computer Science, Nanjing University of Posts and Telecommunications, Wenyuan Road No.9, Nanjing, 210023, Jiangsu, China.
\textsuperscript{2}Mathematics and Computer Science School, Wuhan Polytechnic University, Huanhu Road No.36, Wuhan, 430048, Hubei, China.

\textsuperscript{*}Corresponding author(s). E-mail(s): \texttt{chenwei@njupt.edu.cn};

\textbf{Abstract}

Attackers can use the domain name system (DNS), a crucial service of the network, to carry out harmful activity. The task of detecting malicious domain names has grown in importance in the fight against network crime. The increasing complexity of DNS attack requires new detection methods to deal with new threats. With machine learning and deep learning approaches, malicious DNS detection methods have made considerable strides in recent years. However there are still certain difficulties, such as inflexible feature extraction. To solve this problem, the combination model presented in this paper is based on an enhanced Transformer and a convolutional neural network (CNN). The encoder portion of the Transformer is used to extract domain name features once the network packet has been parsed. Using a multi-head attention mechanism, we pay particular attention to the sequence relationship between the domain name characters. The position relations and word vectors are directly inserted into the word embedding layer using one-hot encoding. They are forwarded to the CNN model for flow-based feature extraction after other traffic information has been analyzed. The features that were extracted by CNN and Transformer are combined into a feature matrix before being categorized, which increases the adaptability of feature extraction. The results of the
experiments show that the proposed method is capable of accurately identifying different categories of malicious DNS with various features.

**Keywords:** DNS security, feature extraction, Transformer, multi-head attention, convolutional neural network

## 1 Introduction

The domain name system (DNS), a crucial fundamental function on the modern Internet, is in charge of translating IP addresses and domain names back and forth. DNS changes domain names that are simple to remember for people into IP addresses that are simple for computers to recognize[1]. As a result, DNS services are essential to many network activities. Nevertheless, DNS services may be misused for a variety of harmful purposes, including the dissemination of malware, the facilitation of communication with Command and Control (C&C) servers[2], sending spam[3], DNS tunnel attack[4] and networking phishing webpage. For instance, a botnet might create a list of domain names using the domain name generating algorithm (DGA), register some of the domains, bind the IP address of the command and control server to create a communication channel, and then execute network attacks. Attackers on the internet create phishing websites by registering domain names that spell popular domain names similarly in order to trick consumers into visiting the page, where they subsequently reveal their own account details[5]. In August 2019, Emotet, one of the most damaging pieces of malware, attacked state and local governments across the United States by sending spam emails using subdomains of multiple top-level domains that were specifically designed to personalize and induce recipients to open malicious document attachments or click on links to download malicious documents[6].

Although the DNS threat is not a new issue, it is still not entirely resolved. Due to the COVID-19 epidemic, more and more people are required to work from home and rely on various cloud services on a daily basis. Cloud services are the target of an increasing number of DNS attacks. According to the EfficientIP and IDC 2021 Global DNS Threat Report, approximately 87% of the organizations surveyed experienced DNS attacks in 2021, 8% higher than the 2020 statistics[7]. Therefore malicious DNS traffic detection has practical significance in countering cyberspace threats, maintaining network stability and ensuring normal service[8].

In the past, filtering domain names with a blacklist or whitelist was widely used to prevent malicious DNS[9][10], but this method could fail to detect dynamically generated domain names. Malicious DNS attack and defense is a long-term confrontation process. With the rapid development of DNS attack methods, machine learning methods have been applied to construct detection
model. Compared to filtering method, machine learning or deep learning methods can be more flexible to detect dynamic changing malicious DNS, achieving effective detection results.

An effective feature engineering process is crucial in machine learning-based methods for enhancing the performance of detection models. Over time, DNS traffic and logs have been used in studies on DNS detection to extract features[11]. Text-based features and flow-based features are popular features. A significant amount of manual effort is required when using machine learning to extract features, demanding a high level of knowledge and machine learning experience on the part of researchers.

There are still difficulties in the feature selection of malicious DNS traffic, despite the extensive work that has been done. The feature extraction system must concentrate on useful features from dynamically changing traffic because different DNS attacks exhibit distinct properties. The manual feature extraction method is inflexible and unable to distinguish between many malicious DNS variants[12]. Furthermore, attackers might be able to quickly evade manually extracted features. The research on malicious DNS detection faces the difficulty of how to construct a model with high detection accuracy and strong flexibility.

Deep learning technology has made remarkable achievements in image processing, speech recognition and other fields, and is widely used in the medical industry, network security and other fields[13]. A powerful advantage of deep learning is feature learning, that is, features are automatically extracted from raw data, and features at higher levels in the hierarchy are formed by the combination of features at lower levels, minimizing manual intervention[14]. Therefore, this paper adopts the deep learning model to extract features automatically including both text-based features and flow-based features. The Seq2Seq[15] is utilized in deep learning model to handle domain names with ambiguous text lengths. It is challenging for the original Recurrent Neural Network-based Seq2Seq model to focus on the long-distance features in the sequence. Because each step’s input depends on its predecessor’s output, the model cannot be parallelized or used efficiently. Even while the Seq2Seq model, which is solely based on Convolutional Neural Networks (CNN), can be implemented in parallel, huge data still makes it challenging to modify parameters. More crucially, lengthy sequence samples cannot be processed using it directly.

We suggest utilizing the Transformer model, a Seq2Seq model with an attention mechanism, to detect malicious DNS. It uses the attention mechanism model alone and has an encoder-decoder structure[16]. It makes advantage of input and output global dependencies to let the model concentrate on important features while ignoring unimportant ones. In addition to investigating long-distance relationships in DNS data, this detection approach can carry out parallel computation for a variety of network traffic. In this study, text-based features are extracted using the Transformer model encoder, and the trigonometric function encoding is swapped out for the one-hot encoding to represent positional data in the sequence. The improved encoder reduces the
computational complexity and improves the detection efficiency. On the other hand, CNN is used to extract flow-based features, and later the two types of features are fused into a feature matrix for model training. The method in this paper focuses on two kinds of features and uses different deep learning models to extract them automatically, which has done good optimization in feature selection and model performance. Compared with the traditional detection methods, the detection efficiency of this model is significantly improved with little manual inference.

In conclusion, the main contributions of this paper are as follows:

1. We create a feature extraction model without the need for labor-intensive human labeling. Transformer encoder is used to extract text-based features from domain names, while CNN model is used to extract features from traffic. The suggested feature extraction and selection technique increases model effectiveness while minimizing manual labeling work.
2. A model called ITransformer_CNN is suggested for the detection of various forms of malicious DNS traffic. To create a new feature matrix, the text-based features and traffic-based features are combined. Depending on various malicious DNS attacks, the ITransformer_CNN model can separately extract various feature matrices. DNS traffic datasets from various circumstances are utilized to assess the detection model’s generalization ability.

The rest of this paper is structured as follows. We briefly introduce the related machine learning based malicious traffic detection work in section 2, especially feature engineering method. Section 3 describes the overall structure design of ITransformer_CNN model. Section 3 introduce the Transformer encoder and CNN model in details. Section 5 analyzes experimental results with CIC-Bell-DNS 2021 datasets. Section 6 makes conclusions.

2 Related work

Over the years, there has been a certain accumulation of research on malicious DNS detection. Researchers have designed many machine learning models to detect them. The key to the performance of these models lies in feature engineering, that is, extracting detection features of distinguishing malicious DNS from benign DNS. The feature selection in previous studies mainly focuses on domain name language feature, DNS traffic feature and auxiliary information feature. Domain name language features usually include domain name character features, n-gram frequency distribution, and semantic level features. n-gram frequency distribution is a language sequence model used in computational linguistics and other fields. Xu et al.[17] input n-gram characters into a deep convolutional neural network and proposed a new malicious domain name classification model (n-CBDC) based on n-gram combining characters. DNS traffic features are usually information captured by parsing DNS traffic, including resolution records, spatial characteristics such as country number,
IP location and time characteristics. Sun et al. [18] conducted a comprehensive analysis of DNS scenarios from multiple perspectives (domain names, clients, attackers), and designed a malicious domain name detection system named DeepDom, which combined DNS scenarios represented as a Heterogeneous Information Network (HIN), which contains different entities such as clients, domains, IP addresses, and accounts for richer information. Samuel Schüppen et al. [19] proposed the domain name detection system FANCI, which detected malicious DNS domain names by monitoring the responses of non-existent domains (NXD). This system extracted 21 features based on structural features, text features and statistical features, and then used random forest or support vector machine to detect NXDs as benign or malicious. The system had high classification accuracy and generalization ability, and was able to identify previously unknown DNS domains.

Suphannee Sivakorn et al. [20] proposed PDNS, a sensor-based DNS monitoring system along with a backend analysis server, which narrowed analysis down to the process level through extensive monitoring of DNS activity and a set of host-based characteristics, instantly detecting malicious processes within compromised hosts. To detect this attack, PDNS extended the monitored DNS activity context and examined the process context that triggers the activity. Experiments were performed using a random forest classifier, which outperforms most previous work.

Peng Chengwei et al. [21] proposed a malicious domain name detection model CoDetector based on the spatial-temporal concomitant correlations among domain name requests. Firstly, a time interval-based space-time accompanying domain name sequence extraction algorithm was proposed, which extracted the domain names with the space-time accompanying relationship from the original DNS traffic. Then the deep learning algorithm was used to map each domain name into a low-dimensional space feature vector, and finally combined the classification algorithm to train the model. The experimental results showed that the CoDetector model could effectively detect malicious domain names.

Machine learning methods need to manually select features before training models. Deep learning models make up for the shortcomings of artificial feature extraction to a large extent. Therefore some researchers use deep learning to directly select useful features for malicious DNS detection, improving the performance of detection models.

Yang et al. [22] proposed a real-time malicious domain name detection system fast3DS. The system adopted a parallel depth-wise convolutional architecture instead of standard convolutional layers, proposing a lightweight global average pooling connection architecture instead of fully connected layers, which can effectively reduce parameters and computation time and improve model detection accuracy with a lightweight attention mechanism. The detection system could achieve accuracy close to the state-of-the-art with significantly reduced parameters, and the processing capability of the system had been significantly improved.
Chen et al.\cite{23} believed that DNS covert channel messages usually showed random characteristics on the FQDN and were not affected by the type of resource records, so they proposed a DNS covert channel detection method based on FQDN and LSTM models. This method first processed the FQDN of the DNS message, removed the second-level and top-level domain names, etc. Then it unified the string length and put the processed string into the LSTM model for detection. The LSTM model contained a hidden layer of 128 units, which was largely guaranteed to contain the original information; finally, the detection results of the LSTM model were filtered by grouping filtering. The grouping method divided DNS traffic into $n$ groups according to the second-level domain name and top-level domain name. The threshold filtering is applied on false positive traffic to further reduce the false positive rate. The experiment also used the CNN model as a comparison. According to the results, it was found that the LSTM model was better than the CNN model, and the accuracy of the LSTM model reached 99.38%.

Ding et al.\cite{24} in order to detect encrypted DNS tunnels, proposed an end-to-end anomaly detection model based on an attention mechanism based on variational autoencoders. By modeling raw traffic sequence data at the traffic level, feature representations are automatically learned using a bidirectional GRU-based and VAE network, and anomalies are detected by reconstruction errors. Furthermore, authors learnt normal traffic through semi-supervised training method to detect unknown types of encrypted DNS tunnel traffic.

To sum up, previous researches mainly have the following problems in feature extraction and model optimization: (1) The key to malicious DNS detection methods lies in feature representation and feature engineering. The features of most previous research are extracted manually. These manual features usually aim at certain types of DNS traffic, which are not effective for the detection of various types of malicious DNS. (2) The model is too single to be flexible, and different types of features are not handled with appropriate models. In this paper, deep learning models are established for the two types of features respectively. The improved Transformer model is used to extract text-based features to obtain the domain name sequence features. The CNN model is used to extract flow-based features at the same time, and then the two features are fused for finally model training. Experiments show that the proposed method has a higher detection rate than existing methods.

3 ITransformer\_CNN Design

This paper proposes a malicious DNS detection model based on improved Transformer and CNN (ITransformer\_CNN). The overall structure of this model is shown in Figure 1, which mainly includes four modules, traffic parsing, data processing, feature extraction & selection and classification. Firstly, Zeek is used to parse traffic packets to obtain domain name text and traffic information in traffic parsing module. Secondly, in data processing module,
some anomalies in DNS traffic data are processed, such as missing values, outliers, data with different magnitude. Thirdly, different features are extracted according to different DNS traffic without manual labelling work by ITransformer_CNN model. Finally, activation functions are used to perform finally DNS traffic classification.

![Fig. 1](image)

**Fig. 1** The overall structure of ITransformer_CNN model.

### 3.1 Traffic Parsing

This module uses Zeek, a network traffic analysis tool, to parse DNS traffic packets. In user behavior analysis, protocol identification and feature extraction play important roles when parsing network traffic. Zeek is a security monitoring tool that finds traces of suspicious activity by deeply inspecting network packets, and can also be used in many traffic analysis models to help extract traffic characteristics\[25\][26]. Zeek parses DNS traffic packets and generates log files, among which DNS logs and connection logs that are crucial for DNS traffic feature extraction. The dns.log is one of the most important log files generated by Zeek, because the DNS protocol is necessary for normal network operation and is generally not blocked by firewalls. Intruders often use DNS protocol to create side channel for nefarious activities, such as carrying command and control traffic, obfuscated or encrypted payload data. The conn.log is a basic log file that provides a lot of information about who is talking to whom, when, how long, and what protocol is used. After converting the two logs files into csv files, we firstly merge them according to the common traffic ID to generate a single csv file containing all the features.
According to past experience and other literature inspiration, we remove several useless features, such as Uniform fields, source IP address destination IP address. Finally there are 23 features left, including domain name features, DNS features, connection features. The domain name itself is input to the improved Transformer model to obtain sequence features. DNS features mainly contain information about DNS queries, such as query type, query class, and RR(resource records). Connection features mainly focus on network connection information, include both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). The feature selection takes into account both the domain name text and traffic information, which are more comprehensive and accurate. The detailed feature description is shown in Table 1.

<table>
<thead>
<tr>
<th>category</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain name</td>
<td>rtt qclass qtype rcode AA TC</td>
</tr>
<tr>
<td></td>
<td>RD RA Z answers TTLs rejected</td>
</tr>
<tr>
<td>DNS features</td>
<td>duration orig_bytes resp_bytes</td>
</tr>
<tr>
<td></td>
<td>conn_state missed_bytes</td>
</tr>
<tr>
<td>connection features</td>
<td>history orig_pkts orig_ip_bytes</td>
</tr>
<tr>
<td></td>
<td>resp_pkts resp_ip_bytes</td>
</tr>
</tbody>
</table>

3.2 Data Processing

In the research of network traffic detection, it is necessary to process the original data. The main problems of the dataset are: missing values, the lack of comparability of different feature magnitudes, the lack of unified standards and definitions. Therefore, it is necessary to convert the data into suitable format for machine learning methods, such as data fusion, data cleaning, data filling, etc. In this paper, after analyzing the data packets to obtain the domain name and traffic information, the data processing of feature set is carried out.

In the field of natural language processing, the length of the sequence text varies, but the sequence text is usually required to keep the same length before being input into the model as a matrix. The domain name sequence text in this paper also needs to be set in uniform length. When the length of domain name is not enough, it is padded with 0. If the domain name is too long, it needs to be truncated.

Flow-based features usually have missing values and some fields have different magnitudes and characters. Since such data cannot be analyzed and trained directly, they need to be processed by data cleaning, character encoding and standardization.

1. Firstly, there may be some anomalies in DNS traffic data, including missing values, outliers. Most fields with missing values are filled. It can be observed that these fields contains mainly one value, so directly filling by the model will not have much impact on the model training, and will not lose key information. Some fields contain a variety of numerical value options, such
as TTLs (the time-to-live interval of DNS query) and answers (the resource
description set in the query answer). If they have missing value, they are
filled with the average value.
2. Secondly, non-numeric fields plays a role in expanding features and are
handled by one-hot encoding. From dataset, we can observe that most non-
numeric fields have limited values range, so one-hot encoding method will
not make the feature matrix too sparse.
3. Finally, in order to eliminate the influence of different order of magni-
tude differences between different eigenvalues, the values are uniformly
normalized. Normalization makes the data scale between [0,1] and makes
the weights between features the same. Common normalization methods
include min-max normalization and z-score normalization. In this paper,
min-max normalization is applied to scale the data to the range between
[0,1] according to min and max of the feature values.

3.3 Feature Extraction & Selection
The feature extraction of DNS traffic is carried out from two aspects, text-
based features and flow-based features. In general, the ITransformer_CNN
model extracts features with improved Transformer and CNN separately, com-
bines them to training model, and makes classification. Considering that the
domain name itself is equivalent to the lexical sequence, the Transformer model
emerging in the field of natural language processing is suitable to extract lex-
ical features. The original position encoding is replaced by one-hot encoding
to directly represent the position information, which reduce the model com-
plexity. Then CNN is used to extract flow-based features and the two features
are fused to train malicious DNS detection model. Since feature extraction
& selection and final classification modules are the main components in the
ITransformer_CNN detection model, we will introduce details in Section 4.

4 ITransformer_CNN Detection Method
4.1 Improved Transformer based Feature Extraction
Transformer abandons the CNN and RNN structures, which are widely used
in deep learning tasks. It only uses the attention mechanism, which reduces
the amount of computation and improves parallel efficiency without degrade
the experimental results, greatly improving the detection performance of the
model. The Transformer model uses the encoder-decoder architecture, where
the encoder maps the input sequence \((x_1, x_2, \ldots, x_n)\) to continuous representa-
tions \((y_1, y_2, \ldots, y_n)\), and the decoder generates an output sequence
\((z_1, z_2, \ldots, z_n)\) at each moment. Since text generation is not required in our
detection model, only the encoder structure of Transformer is reserved. The
encoder contains N layers, and each layer contains two sub-layers, namely
multi-head attention layer and fully connected network layer. Each sub-layer
performs residual addition and layer normalization [27]. Before the input information enters the encoder, position encoding is performed first. Since the Transformer abandons the recursion in the RNN and cannot capture the sequence order information, the relative or absolute position information of the sequence must be expressed by position encoding. The structure of the improved Transformer model is shown in Figure 2.

![Transformer model structure](image)

**Fig. 2** Transformer model structure.

### 4.1.1 Position Encoding

General deep learning models such as CNN and RNN can capture position information by themselves. The position information has little effect on the results and is not necessary to put into models. However, the Transformer cannot capture position information, so the position information needs to be put into the model together with the word vector. The position encoding in the traditional Transformer model uses the trigonometric function encoding
method to extract the absolute and relative position information of the vocabulary sequence. Since only the absolute position information of the characters in the domain name is needed when extracting the character features of the domain name, relative position information has little effect on the model. So the absolute position information is directly represented by one-hot encoding. Assuming that the length of the domain name after the unified length is \( s \), then the matrix dimension after the one-hot encoding of the domain name sequence is \( s \times s \), as shown in Figure 2. The diagonal positions of this matrix are all set to 1, and the other positions are 0. The dimension of embedding matrix is \( s \times d \), and the dimension after connecting the embedding matrix and the one-hot matrix is \( s \times (s + d) \), which is put to the encoder layer of the Transformer. One-hot encoding is used instead of triangular encoding, which reduces the complexity of the model without affecting the effect of the model.

4.1.2 Multi-Head Attention Mechanism

The self-attention mechanism used in Transformer is a variant of the attention mechanism, which reduces the dependence on external information. It is especially good at capturing the internal correlation of data. Assuming that the input matrix is \( E = [x_1, x_2, ..., x_s] \), the dimension of \( E \) is \( s \times (s + d) \). Each input \( x_i \) can be mapped into a query vector \( q \), a key vector \( k \), and a value vector \( v \). The dimensions of \( q \) and \( k \) are \( d_k \), and the dimension of \( v \) is \( d_v \), and for the convenience of calculation, they will be performed linear mapping respectively to obtain the matrices of \( Q, K, \) and \( V \). Calculated as follows:

\[
Q = W_q E \\
K = W_k E \\
V = W_v E
\]

Where \( Q \in \mathbb{R}^{s \times d_k}, K \in \mathbb{R}^{s \times d_k}, V \in \mathbb{R}^{s \times d_v} \), \( W_q, W_k, W_v \) are three mapping parameter matrices. Calculate dot product between \( Q \) and \( K \), and divide by \( \sqrt{d_k} \), and then apply the softmax function to calculate the weight. The formula is as follows\[16\]:

\[
Attention(Q, K, V) = softmax\left(\frac{Q^T K}{\sqrt{d_k}}\right)V
\]

It is not enough to do such weighting operation on \( Q, K, \) and \( V \) only once. Multi-head attention mechanism was first proposed in Transformer. The input \( q, k \) and \( v \) are divided into \( h \) heads, so in each head \( d_k = d_v = (s + d)/h \). multiple heads are linearly mapped to obtain multiple sets of initialization matrices \( Q, K, \) and \( V \). After obtaining matrices, it performs attention calculation separately, connects multiple attention results and linearly maps the connected results again. The calculation formula is as follows:

\[
MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^o
\]
where \( \text{head}_j = \text{Attention}(QW_j^Q, KW_j^K, VW_j^V) \)

Among them, \( \text{head}_j \) is the output of the \( j \)-th head, \( W_j^Q \in R^{(s+d) \times d_k} \), \( W_j^K \in R^{(s+d) \times d_k} \), \( W_j^V \in R^{(s+d) \times d_v} \)[16]. The parameter \( W \) of the linear transformation of \( Q \), \( K \), and \( V \) is different each time. The difference of the multi-head attention mechanism is that it performs multiple calculations instead of just one calculation, which has the advantage of allowing the model to learn relevant information in different representation sub-spaces.

### 4.2 CNN based Feature Extraction

CNN is widely used in computer vision, machine translation, traffic detection and other fields. It mainly has three layers: convolution layer, pooling layer, and fully connected layer. The CNN model in this paper uses two layers of convolution and two layers of maximum pooling. In the convolution layer, the convolution kernel is used to extract and combine the local area features. The function of the pooling layer is to downsample. The matrix after convolution is divided into several matrices of the same size, and then calculated the average and maximum value of the features. The pooling layer reduce the parameters while retaining the significant features, prevent overfitting, and improve the generalization ability of the model. The fully connected layer can classify the features after convolution and pooling, which obtains the classification result by adjusting the weights of the network. In our CNN model, a Dropout layer is added before the fully connected layer. During the deep learning training process, the neural network training units are temporarily removed from the network according to a certain probability. So each data batch processing is being trained with different networks[28]. Dropout layer is used to reduce overfitting and improve the generalization ability of the network.

### 4.3 Detection Algorithm

In this module, the processed domain name sequence is first put into the word embedding layer, which is a standard part of the natural language processing field. Each character is converted into a word vector, combined into a word embedding matrix, and then put into the Transformer model. The position encoding part directly uses one-hot encoding. For each domain name sequence, a sparse matrix with dimension is generated to describe the position of each character in the sequence. A new embedding matrix is obtained after the position encoding information is connected. The global maximum pooling is performed to reduce the dimension of matrix, which is then put to the fully connected layer. The processed traffic features are filtered by one-dimensional convolution using the CNN model, and then downsampled by global maximum pooling. After that, the fully connected layer is used to map the previously extracted features to the sample space, which also plays a dimensionality reduction role. Finally, the two features are fused, and then classified through the fully connected layer. The specific steps of the model algorithm are shown in Algorithm 1.
Algorithm 1 ITransformer_CNN detection algorithm

Input: Domain name sequence \( s = (s_1, s_2, ..., s_n) \), \( H \) = the number of multiple attention heads, \( L \) = the number of encoding layers, inputB = traffic information matrix
Output: predicted probabilities for each category

1. Use ITransformer to extract text-based features
   1.1: Use Embedding layer to transform \( S \) into word embedding matrix \( E \)
   1.2: EmbeddingPosition = [ ]
   1.3: For \( i=1 \) to \( n \)
        1.3.1: Create a array PE filled with n zeros
        1.3.2: PE[\( i \)]=1
        1.3.3: Add PE in embeddedPosition
   1.4: End for
   1.5: Concat E and embeddedPosition to M
   1.6: For \( el=1 \) to \( L \)
        1.6.1: OutputMH=[ ]
        1.6.2: For \( j=1 \) to \( H \)
               1.6.2.1: Perform self-attention calculation head\(_j\) for multiple spatial matrices of E
               1.6.2.2: Concat head\(_j\) to outputMH
        1.6.3: End for
        1.6.4: Use outputMH and E to perform layer normalization outputLN1
        1.6.5: Use the full connection layer to integrate outputLN1
        1.6.6: Use outputDS and outputLN1 to perform layer normalization outputsLN2
        1.6.7: M= outputsLN2
   1.7: End for
   1.8: OutputA=M
   1.9: Use pooling layers for dimensionality reduction of outputA
   1.10: Use Dropout layer to reduce overfitting of outputA
   1.11: Use the full connection layer to integrate outputA

2. Use CNN to extract flow-based features
   2.1: Use convolution layer to extract the feature outputB of inputB
   2.2: Use pooling layers for dimensionality reduction of outputB
   2.3: Use Dropout layer to reduce overfitting of outputB
   2.4: Compute outputB one-dimensional
   2.5: Use the full connection layer to integrate outputB

3. Concatenate the two features
   3.1: Concat outputA and outputB to outputC
   3.2: Use softmax to compute the classification probability of outputC
   3.3: Return outputC
   3.4: End

5 Experiment

5.1 Dataset

CIC-Bell-DNS 2021 dataset is used in this paper[29]. It is a large-scale DNS dataset generated and published by a collaborative project from the Canadian Cyber Security Institute and the Cyber Threat Intelligence Centre (CTI). Using data from 1 million benign domains and 51,453 known malicious domains in publicly available data sets, the researchers generated 400,000 benign samples and 1,301 malicious samples by reproducing real-world scenarios with frequent benign traffic and multiple malicious domain types. The whole dataset includes one kind of benign samples and three kinds of malicious samples, which are spam, phishing and malware. After all traffic packets are parsed with
Zeek, the parsed traffic samples are down-sampled to keep balanced. The proportions of normal samples and malicious samples in the experimental dataset are listed in Table 2. After shuffling the data, we divide the training set and test set according to the ratio of 3:1.

<table>
<thead>
<tr>
<th>type</th>
<th>number</th>
<th>proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>spam</td>
<td>4337</td>
<td></td>
</tr>
<tr>
<td>phishing</td>
<td>4337</td>
<td>39%</td>
</tr>
<tr>
<td>malware</td>
<td>4337</td>
<td></td>
</tr>
<tr>
<td>benign samples</td>
<td>20000</td>
<td>61%</td>
</tr>
</tbody>
</table>

5.2 Experimental environment and evaluation indicators

In order to train and test ITransformer_CNN proposed in this paper, the experiment is carried out in the Linux system environment, with Intel™Xeon™Gold 5218 CPU 2.30GHz, 128G memory and RTX4000*2 8G GPU. The deep learning model is built using tensorflow 2.6 and keras 2.6 in python3.7.

In order to evaluate the detection effect of the malicious DNS binary classification model, this paper adopts the accuracy rate (A), recall rate (R), and F1 value as evaluation indicators. The F1 value is a harmonic mean of precision (P) and recall rate (R). Confusion matrix is the basic evaluation index. Four basic indicators are calculated by the confusion matrix: TP represents the number of positive samples that are correctly classified; FP represents the number of sample, which is wrongly predicted to be positive but is actually negative; FN represents the number of sample, which is wrongly predicted to be negative but is actually positive; TN represents the number of correctly classified negative samples. Then, according to these four basic indicators, the evaluation indicators are calculated as follows:

\[
A = \frac{TP + TN}{TP + TN + FP + FN}
\]

\[
P = \frac{TP}{TP + FP}
\]

\[
R = \frac{TP}{TP + FN}
\]

\[
F1 = 2 \frac{P \times R}{P + R}
\]

5.3 Performance result analysis

The overall structure of the model has been introduced in the third section. In deep learning, the setting of different parameters has a certain influence

1https://github.com/clcw-yy/ITransformer-and-CNN
on the quality of the model. In this experiment, some parameter settings are carried out based on past experience. In ITransformer, the encoder is set to 2 layers, the word embedding dimension is set to 128, and the maximum sequence length is 40. Therefore the input dimension of the encoder is calculated as 168 by adding the word embedding dimension and the sequence length. After the two-layer encoder, the global average pooling layer is used to reduce the dimension of features, which reduces network model parameters. In order to avoid model overfitting, a dropout layer is also added and the dropout ratio is 0.3. The number of neurons in the fully connected layer is 64.

In ITransformer, the multi-head attention mechanism is used to extract and splice important information of multiple subspaces. The multi-head attention mechanism is similar to a sampling technique, among that each attention head is a sample. The greater each head discriminates, the more viewing angle it has. We observe the accuracy of the model by changing the number of attention heads. Since the space needs to be divided according to the number of heads, the set number of heads must be divisible by ITransformer’s input dimension. The accuracy of the model with different numbers of heads is shown in Figure 3. It can be seen that with the increase of the number of heads, the accuracy of the model increases significantly. After the number of heads exceeds 8, the accuracy rate does not change significantly, and the effect tends to be flat or even decline. According to the results in Figure 3, we set the head number to 8.

In CNN, two-layer convolution and two-layer maximum pooling are used in this experiment. We set the filter size of the first convolution layer to 64, the filter size of the first convolution layer to 32, the convolution kernel size to 2, the maximum pooling layer size to 3, the stride to 3, and the fully connected layer to 64. The relu function is used as activation function.

Finally, the fully connected layer output of the two models was connected by Concatenate function, then the fully connected layer with 4 units was used for multiple classification, and the activation function was softmax to obtain the detection result. In the final training, the hyperparameters have also been evaluated many times to select the best value. The learning rate is set to 0.0001 while the optimizer uses Adam as the optimization algorithm, which is computationally efficient and can better handle noise samples and sparse
gradients. The overall model is iterated for several times, and the loss rate changes as the number of iterations increases, as shown in Figure 4. The loss rate is almost unchanged after the epoch is greater than 80. In order to prevent over-fitting of the model, the epoch is set to 80.

Fig. 4 The loss rate changes when epoch increases.

After testing the test set, the result of multiple classification output. Figure 5 shows the four types of confusion matrix of heat map, can be seen that the prediction accuracy of the model in this paper is higher. The spam samples are all correctly detected with 0 false positive and 0 false negative. However the number of false negative malware samples was 153, and the model did not accurately distinguish between malware and phishing. In general, the detection effect of this model is good, and it has good effect on the four types.

Fig. 5 Test set confusion matrix heat map.

5.4 Comparison experiment with different encoding methods

As mentioned above, the Transformer model itself does not have the ability to learn word order information like a recurrent neural network, and needs to actively input word order information to the model. Then, the original input
of the model is the word vector without word order information. Position encoding needs to combine the word order information and word vector to form a new representation input to the model, so that the model has the ability to learn word order information. For the position encoding, ITransformer has made some improvements, changing the original trigonometric function encoding to one-hot encoding. This paper conducts experiments to compare the effects of the model using triangular encoding and one-hot encoding. The results are shown in Table 3.

<table>
<thead>
<tr>
<th>position coding</th>
<th>A</th>
<th>detection time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>trigonometric encoding</td>
<td>0.9590</td>
<td>4972.70</td>
</tr>
<tr>
<td>one-hot encoding</td>
<td>0.9684</td>
<td>3440.60</td>
</tr>
</tbody>
</table>

It can shows that the encoding method used in this experiment is better than the traditional trigonometric function encoding method, and the accuracy rate is improved to 96.84%. Because one-hot encoding does not require a lot of computation, the detection time is reduced by about 25 minutes.

### 5.5 Comparative experiments with other models

The proposed ITransformer model is compared with LSTM model, which is often used to process sequence data. Both models only extract text-based features for domain name. In addition, the ITransformer.CNN is compared with a single ITransformer to prove the superiority of extracting features using the combined model. Finally, ITransformer.CNN and LSTM.CNN are compared to further verify the effect of the proposed method. The accuracy changes of the four models are shown in Figure 6 respectively.

![Fig. 6 Change in val_accuracy of different models.](image)

It shows that when the epoch is less than 50, ITransformer’s val_accuracy is always smaller than LSTM’s. When the epoch is greater than 65, ITransformer’s val_accuracy is obviously higher than LSTM’s. At this time, LSTM’s accuracy reaches the highest, about 0.9070. The val_accuracy of ITransformer reaches the highest value of 0.9288 when the epoch is 80, and then leveled off.
Similarly, ITransformer_CNN has a higher effect than LSTM_CNN when the epoch is greater than 50, and the val_accuracy of ITransformer_CNN increase 2.37 percentage points, which verifies the effect of ITransformer. It can be seen that the combined model is better than the single model, which further illustrates the effect of feature extraction using different deep learning models.

Mahdavifar et al.[29] proposed a feature extraction method for malicious DNS detection method with developing 32 features, including text-based, DNS statistical-based, and third party-based features, then KNN, SVM, MLP, GNB and LB are selected for classification. The classification models used in the literature are compared with that in this paper, and the best, LR, MLP and KNN, are selected here. The model configuration is shown in Table 4 and the comparison results are shown in Table 5 and Table 6.

As we can seen from the two table, the Recall and F1 value of our model are both over 90%. In Table 5, the Recall of the model in this paper is not as good as KNN in detecting benign samples, but the other three categories of detection are higher than KNN, especially the malware and phishing. The Recall of the proposed model 7.72% higher than KNN in detecting malware, and 5.93% higher than KNN in detecting phishing. In Table 6, the F1-score detected by the proposed method on the four types is higher than that of the other three models. Compared with KNN, the F1 value by the proposed model in detecting malware is increased by 2.48%, and the recall rate detected by the proposed model on phishing is increased by 4.80%. It can be seen from the two table that the detection effect of MLP and LR algorithm is not good, especially for malware and phishing, which further indicates that the detection effect of the proposed model on the four categories is excellent, and verifies that the proposed model can adapt to the change of traffic features in the face of multiple types of malicious DNS.

Table 4 configuration of machine learning classifiers for comparison

<table>
<thead>
<tr>
<th>classifier</th>
<th>configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN</td>
<td>n_neighbors=3</td>
</tr>
<tr>
<td>MLP</td>
<td>hidden_layer_size(30,20,10), activation=relu,</td>
</tr>
<tr>
<td></td>
<td>solver=adam, learning_rate_init=0.001</td>
</tr>
<tr>
<td>LR</td>
<td>solver=lbfgs, penalty=l2</td>
</tr>
</tbody>
</table>

Table 5 Comparison of Recall between the proposed model and KNN, MLP and LR

<table>
<thead>
<tr>
<th>category</th>
<th>the proposed model</th>
<th>KNN</th>
<th>MLP</th>
<th>LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>benign</td>
<td>0.97401</td>
<td>0.9770</td>
<td>0.9535</td>
<td>0.9554</td>
</tr>
<tr>
<td>spam</td>
<td>1</td>
<td>0.9953</td>
<td>0.8997</td>
<td>0.3611</td>
</tr>
<tr>
<td>malware</td>
<td>0.9095</td>
<td>0.8443</td>
<td>0.5514</td>
<td>0.0229</td>
</tr>
<tr>
<td>phishing</td>
<td>0.9747</td>
<td>0.9201</td>
<td>0.07608</td>
<td>0.3548</td>
</tr>
</tbody>
</table>
Table 6 Comparison of F1-score between the proposed model and KNN, MLP and LR

<table>
<thead>
<tr>
<th>category</th>
<th>the proposed model</th>
<th>KNN</th>
<th>MLP</th>
<th>LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>benign</td>
<td><strong>0.9829</strong></td>
<td>0.9816</td>
<td>0.9167</td>
<td>0.7894</td>
</tr>
<tr>
<td>spam</td>
<td>1</td>
<td>0.9787</td>
<td>0.8910</td>
<td>0.4616</td>
</tr>
<tr>
<td>malware</td>
<td><strong>0.9005</strong></td>
<td>0.8603</td>
<td>0.6309</td>
<td>0.0436</td>
</tr>
<tr>
<td>phishing</td>
<td><strong>0.9251</strong></td>
<td>0.8827</td>
<td>0.7932</td>
<td>0.4373</td>
</tr>
</tbody>
</table>

5.6 Evaluation results using other datasets

In order to verify the generalization ability, we use DataCon 2019 Security Analysis Competition DNS datasets to evaluate the ITransformer_CNN model. The dataset internal data is a network traffic bypass mirroring device in a period of time Tsinghua campus network captured packets, including benign DNS traffic and DGA traffic. Similarly, Zeek is used to parse the data packet and remove DNS traffic containing duplicate domain names to obtain 60178 pieces of data, among which DGA traffic accounts for 12500 pieces. After data processing, the proposed model is used to extract features and identify DGA domain names. The result show that It also gives good detection results on this dataset. As shown in Table 7, the Recall of the benign and DGA traffic are 97.56% and 96.74% respectively, and the F1 values of two categories are both above 95%. The detection results on Datacon 2019 dataset indicate that the ITransformer_CNN model can also perform well on other dataset and has strong generalization ability.

Table 7 The detection results on DataCon 2019 dataset

<table>
<thead>
<tr>
<th>category</th>
<th>A</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>benign</td>
<td>0.9761</td>
<td>0.9756</td>
<td>0.9793</td>
</tr>
<tr>
<td>DGA</td>
<td>0.9568</td>
<td>0.9674</td>
<td>0.9510</td>
</tr>
</tbody>
</table>

6 Conclusion

Aiming at the challenges in today's DNS detection research, this paper proposes a combined model based on an improved Transformer and CNN, which extracts text-based domain name sequence features through Transformer, and extracts flow-based traffic features through CNN. The two features are fused together to obtain a new feature matrix, and then used for classification. In addition, the position encoding method in Transformer is changed from trigonometric encoding to one-hot encoding and only the absolute position information in the sequence is reserved. Experimental results show that this method can improve the model detection efficiency and reduce the computation complexity.

The datasets used in this paper are collected from network traffic in the real world. ITransformer_CNN models shows good generality on different datasets. In the future, we plan to deploy ITransformer_CNN model for online traffic

---

2https://datacon.qianxin.com
detection, which requires high generalization ability of the model. At the same
time, due to the small proportion of abnormal traffic in online environment,
the unbalanced sample problem should be handled in online deployment. Sub-
sequent work mainly focuses on the improvement of model with unbalanced
sample and the enhancement of generalization ability.

Declarations

Ethics Approval
This work does not contain any studies with human participants or animals
performed by any of the authors.

Conflict of Interest
I declare that the authors have no competing interests as defined by Springer,
or other interests that might be perceived to influence the results and/or
discussion reported in this paper.

Data Availability
We used open datasets from Internet and have given related link or cita-
tion in the paper. The code for reproducing our main results is available at
https://github.com/clcw-yy/ITransformer-and-CNN

Author Contribution
* Wenyue Zhang: Methodology, Validation, Software, Investigation, Writing-
Original draft, Data curation.
* Wei Chen: Conceptualization, Supervision, Resources, Writing - Review and
Editing preparation.
* Zhen Zhang: Data curation
* Fan Zhang: Conceptualization, Visualization
* Lifa Wu: Funding acquisition

Funding
This work is supported by the National Key Research and Development
Program of China under Grant No.2019YFB2101704.

Consent to publish
Not applicable.
Acknowledgement

We would like to thank our anonymous reviewers for their valuable comments and suggestions. This work is supported by the National Key Research and Development Program of China under Grant No.2019YFB2101704.

References

[12] Ryan R Curtin, Andrew B Gardner, Slawomir Grzonkowski, Alexey Kley- 
menov, and Alejandro Mosquera. Detecting dga domains with recurrent 
normal networks and side information. In Proceedings of the 14th Inter- 
national Conference on Availability, Reliability and Security, pages 1–10, 
2019.

Li. Influencing factors analysis in pear disease recognition using deep 
2021.

and cost-efficient offloading policy for mobile cloud computing against 

line between linguistic generalization and failure in seq2seq-attention 

Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention 
is all you need. Advances in neural information processing systems, 30, 
2017.

names generated by dgas based on semantic representation and deep 

[18] Xiaooqing Sun, Zhiliang Wang, Jiahai Yang, and Xinran Liu. Deep- 
dom: Malicious domain detection with scalable and heterogeneous graph 

[19] Samuel Schüppen, Dominik Teubert, Patrick Herrmann, and Ulrike 
Meyer. FANCI: Feature-based automated NXDomain classification and 
intelligence. In 27th USENIX Security Symposium (USENIX Security 

[20] Suphannee Sivakorn, Kangkook Jee, Yixin Sun, Lauri Korts-Pärn, 
Zhichun Li, Cristian Lumezanu, Zhenyu Wu, Lu-An Tang, and Ding Li. 
Countering malicious processes with process-DNS association. In NDSS, 
2019.

[21] Chengwei Peng, Xiaochun Yun, Yongzheng Zhang, and Shuhao Li. Detect- 
ing malicious domains using co-occurrence relation between DNS query. 

[22] Luhui Yang, Guangjie Liu, Jinwei Wang, Huiwen Bai, Jiangtao Zhai, and 
Yuwei Dai. Fast3ds: A real-time full-convolutional malicious domain 
name detection system. Journal of Information Security and Applications, 

[23] Shaojie Chen, Bo Lang, Hongyu Liu, Duokun Li, and Chuan Gao. DNS 
covert channel detection method using the LSTM model. Computers & 

[24] Shuai Ding, Daoqing Zhang, Jingguo Ge, Xiaowei Yuan, and Xinhui


