ER-ERT:A Method of Ensemble Representation Learning of Encrypted RAT Traffic

Yijing Zhang*† , Hui Xue*† , Jianjun Lin* †, Xiaoyu Liu* †, Weilin Gai ‡ § **, Xiaodu Yang* †, Anqi Wang¶, Yinliang Yue‖ ††, and Bo Sun** ††
*Institute of Information Engineering, Chinese Academy of Sciences;
† School of Cyber Security, University of Chinese Academy of Sciences;
{zhangyijing, xuehui, linjianjun, liuxiaoyu, yangxiaodu}@iie.ac.cn;
‡Institute of Software, Chinese Academy of Sciences;
§ School of computer science and technology, University of Chinese Academy of Sciences;
¶Liaoning University of Technology;silverbullet_conan@163.com;
‖Zhongguancun Laboratory, Beijing, P.R.China;yueyl@zgclab.edu.cn;
**National Computer Network Emergency Response Technical Team/Coordination Center of China;
{gwl,sunbo}@cert.org.cn;

***Abstract*—Remote Access Trojan (RAT) is one of the major threats to today's network environment. It is a class of malware frequently used by hacking collectives to monitor victims' actions and steal personal information in targeted computers. Traditional machine learning algorithms have been widely used to detect malicious encrypted RAT traffic. Traditional machine learning algorithms rely deeply on expert experience, and it is difficult for current traffic classification models to design effective handcraft features. Deep learning methods have been introduced in recent years to generate representations from raw network traffic data automatically. Previous deep learning-based malicious traffic detection methods generate representations from flow sequences or packet payload bytes. None of these methods simultaneously learn embeddings from flow sequence and packet payload bytes. Thus, we propose a novel ensemble model to draw fine-grained and multi-angle traffic representations for RAT traffic. The model extract (1) temporal features with convolution neural network (CNN) and the Reproducing Kernel Hilbert Space (RKHS) embedding method to model network flow sequence, (2) spatial features with autoencoder and bidirectional gated recurrent unit (Bi-GRU) network to model packet payload bytes, and (3) some stage-based attributes to enhance the identification ability of RAT traffic behaviors. According to the experimental result, our approach achieves better performance than previous works with a precision rate of 97.0% and a recall rate of 96.5%.**

***Index Terms*—Encrypted Malware Detection, RAT Traffic, Ensemble Learning, Deep Learning**

I. Introduction

Nowadays, Trojan has become one of the most widely used malicious programs in our network environment. Remote Access Trojan (RAT) is a kind of Trojan that allows malicious attackers to control the system and access the victims' information by opening a backdoor in the users' system. Typical functions of RAT include process monitor, command execution and file transfer, etc. RAT can cause great harm to the security of network systems and damage personal property, so it is necessary to specify the malicious activities of RAT to prevent our computers from further information leakage or attack from adversaries. DPI-based detection through string pattern matching is a popular method to identify RAT traffic behavior in industry circumstances. In today's network environment, most network traffic is encrypted, so the actual contents of IP packet payloads are invisible. Since most malware, including RAT, encrypts their network flow for better concealment, the traditional DPI-based detection methods are significantly challenged and usually lose their functionality.

In recent years, researchers have tended to use machine learning methods to tackle the identification problem of malicious encrypted traffic. Traditional machine learning (TML) algorithms, such as Support Vector Machine (SVM) and Random Forest, are applied to classify network flow with artificial traffic features like bytes number or interval gap between packets [1–4]. TML algorithms rely deeply on expert experience, and the feature set design needs lots of human effort. The performance of the trained model will become unstable when transferred into a different environment [5]. To reduce the dependence of the model on feature engineering and realize an end-to-end traffic flow classification or malicious traffic detection, deep learning (DL) methods are introduced to distill traffic features and identify specific network behaviors automatically [6–11]. These DL-based methods distill features from raw network traffic flow instead of using a DL network to learn representation from the artificial features, i.e., the model in [12], which still relies on handcrafted features to guarantee the performance of the method. However, these previous methods have ignored the heterogeneity of network traffic data. They have tended to learn representation from flow sequences or packet payload bytes. None of these methods model flow sequence and packet payload bytes simultaneously. Hence, information loss and compression still exist. These methods fail to detect RAT traffic accurately, according to our experiments. The issue of lacking representative information

†† The first corresponding author is Bo Sun. The second corresponding author is Yinliang Yue.
ISBN 978-3-903176-57-7

for encrypted malware detection has yet to be well resolved.

For malicious RAT traffic detection, improving the stability and accuracy of the models is the main target. Adversaries have developed various evasion techniques by perturbing flow features or packet content features, reducing the amount of usable information for malware identification [13]. Generating embeddings from more flow perspectives is useful for improving the stability and accuracy of the malicious RAT traffic flow detection by fully exploiting the usable information in the flow. Thus, we employ an ensemble representation learning method to learn embeddings from the flow sequence and packet payload. We distill features from the packet payload with the bidirectional gated recurrent unit (Bi-GRU) and autoencoder. Moreover, we apply the sliding-window technique to explore the impact of neighboring packet dependencies on identifying RAT network traffic. These payload-relevant features reveal much information about the encryption suite, security degree, and packet-to-packet interrelationship, i.e., spatial features. In the meantime, we learn representations from the RAT network flow sequence by transforming the raw partial sequence of a flow to a multi-channel image using the Reproducing Kernel Hilbert Space (RKHS) embeddings [14]. The convolution neural network (CNN) generates a low-dimensional feature representation from the multi-channel image. These flow-related features reveal the network behavioral pattern changing with time, i.e., temporal features. Based on observation and statistical analysis, the RAT flow can be divided into three stages due to their function and arrival time [15]. We extract different features from different stages to model the traffic flow more precisely. These staged features are statistic-based, i.e., statistical features. We aggregate all the learned embeddings as the final representation to improve RAT network traffic detection accuracy.

In this paper, we propose the Ensemble Representation model of Encrypted RAT Traffic (ER-ERT), which provides a robust and accurate network feature extraction method for encrypted RAT command and control (C&C) communication traffic identification.

In summary, our main contributions are as follows:

- We propose the model ER-ERT to distinguish encrypted RAT traffic from benign traffic. The ER-ERT model flow sequence and packet payloads bytes in a network flow simultaneously by learning embeddings from both of them with several deep neural networks. ER-ERT can fully exploit the information contained in network flow.
- We designed several stage-based statistical features as distinctive characteristics for RAT traffic detection to complement the features distilled from flow sequence and packet payloads bytes. We find traffic of normal applications differs from RATs on these stage-based features.
- According to our experimental results, ER-ERT achieves a satisfactory performance, with 97.2% precision rate and 98.0% recall rate on the open benign dataset and RAT traffic dataset, and 96.8% precision rate and 97.2% recall rate on the self-collected benign dataset and RAT traffic dataset, outperforming the state-of-the-art methods, showing the effectiveness our model. Meanwhile, ER-ERT keeps a high detection precision of 97.0% and a nice recall rate of 96.5% when trained on the open benign dataset while tested on the self-collected benign dataset, showing the stability of our model.

The remainder of this paper is organized as follows: Section 2 presents an overview of related work in the area. Section 3 gives a detailed description of our model. The experimental setup and the experimental results are depicted in Section 4. Section 5 presents conclusions and future work.

II. Related work

In this section, we provide an overview of the most important methods of traffic classification and malicious traffic detection. In particular, we can categorize these approaches into four main categories as follows: (1) DPI-based approach, (2) TML-based approach, (3) DL-based approach, and (4) graph-based approach.

A. DPI-based approach

DPI-based traffic classification approaches are commonly used in industry. Finsterbusch et al. [16] summarized the current main DPI-based traffic classification methods. Most DPI-based approaches use predefined patterns like regular expressions as signatures to classify traffic flow [17] [18]. The generation of signatures is based on the analysis of information available in the application layer payload of packets. Thus, DPI-based traffic classification methods are unsuitable for abnormal encrypted traffic identification because the application layer payload is encrypted without decryption.

B. TML-based approach

To deal with the problem of malicious traffic identification, TML-based approaches are often used with a careful-designed feature set. Gezer et al. [1] adopted a behavior-based Random Forest model that employed artifacts created by the malware during the dynamic analysis of TrickBot malware samples. Aljawarneh et al. [2] proposed a hybrid model based on optimal network traffic characteristics. Aljawarneh et al. first chose 41 relevant traffic features and used the Information Gain to reduce the feature set size to 8. Anderson et al. [3] took flow-level features such as flow metadata, packet length distributions, and time distributions as joint features to identify encrypted malware traffic. In contrast with Anderson et al., Stergiopoulos et al. [4] used fewer features to classify encrypted traffic with the algorithm classification and regression tree (CART). Stergiopoulos et al. suggested that five cleverly designed side-channel features were enough for detecting multiple types of malicious traffic, including packet size, payload size, payload ratio, ratio to the previous packet, and time difference.

In conclusion, feature engineering is essential to guarantee the performance of TML-based approaches. However, it is costly with the requirements of careful engineering and considerable domain expertise. Besides, with the rapid updating rate of RAT, the predefined fixed features lose their effectiveness

very quickly, resulting in the decline of malicious traffic detection accuracy [19].

C. DL-based approach

In recent years, DL has emerged as a method to automatically generate feature representations from raw network traffic data to realize end-to-end learning. Compared with TML-based methods, DL-based learning methods do not require the design of manual features. Many previous works have used packet payload bytes with DL methods for malware detection or traffic classification. Wang et al. [6] converted the first 784 bytes in a flow into a grey image for further classification with CNN. Similarly, Lotfollahi et al. [7] used the bytes in a flow to classify encrypted traffic. The model "Deep Packet" used CNN or stacked autoencoder (SAE) to extract features from packet payloads and classify the network traffic into different applications. Liu et al. [8] proposed the model BGRUA, which utilized a Bi-GRU and attention mechanism to realize HTTPS traffic classification. In [9], the multimodal multitask DL method DISTILLER for traffic classification was proposed. Two types of input data were fed to the DISTILLER: (1) the first N_b bytes of transport-layer payload and (2) informative protocol header fields of the first N_p packets. Although DISTILLER is a multimodal approach, it models only N_b bytes of payload without learning embeddings from the flow sequence.

The above methods use DL to automatically generate embeddings from the packet payload but ignore learning representation from the raw flow sequence. The network behavior characteristics contained in the flow sequence are also useful information for identifying malware traffic. Shapira et al. [10] proposed the model FlowPic, which transformed a series of packet sizes in each flow into two-dimensional grey images and implemented image classification with CNN. Liu et al. [11] proposed a model called FS-Net, which used an autoencoder to learn the hidden representation from the packet length sequence. The autoencoder consists of a 2-layer Bi-GRU as its encoder and a 2-layer Bi-GRU as its decoder. These methods generate embeddings from the flow sequence but ignore learning representations from packet payloads.

D. Graph-based approach

In addition to TML-based and DL-based methods, researchers have used graph-based methods to give a new way to look at network traffic via connected graphs. Busch et al. [20] extracted a flow graph where the nodes correspond to endpoints in the network and edges represent communication between these endpoints. Statistical features, including mean and standard deviation of the packet length and minimum and maximum interarrival time of the packets, were used to generate the flow graph. Fu et al. [13] proposed the model ST-Graph. Artificial features, including the TLS version and the supported cipher suites, and some statistical information, such as the number of packets and bytes within a stream, are used for graph embedding. Neither approach automatically generates feature representations from network traffic data. Thus, careful feature engineering is still necessary before the construction of graphs. The dependence on feature engineering is costly and could lead to the loss of useful information.

III. Description of ER-ERT framework

A. Overall Architecture

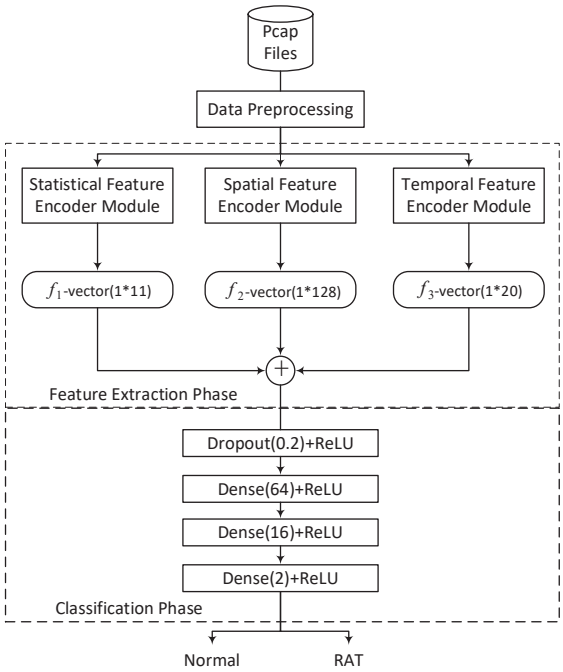


Fig. 1: The overall architecture of ER-ERT. The f_1-vector, f_2-vector, and f_3-vector indicate the embeddings generated by the STA, the SPA, and the TEM, respectively.

Our proposed model ER-ERT framework is illustrated in Fig. 1. The model generates a fine-grained and multi-angle traffic representation for RAT traffic identification. The ER-ERT can be divided into three parts: *Data preprocessing phase, Feature extraction phase*, and *Classification phase*. The *Data preprocessing phase* changes the raw flow data into an ideal form for feature extraction through five steps, including TCP handshake packet removal, data link header removal, zero padding, etc. The *Feature extraction phase* consists of the Statistical Feature Encoder module (STA), the Spatial Feature Encoder module (SPA), and the Temporal Feature Encoder module (TEM). The *Classification phase* consists of one dropout layer and three full connection layers to realize traffic classification.

Network traffic in Pcap files is segmented into flow f_i according to 5-tuple <Source IP, Destination IP, Source Port, Destination Port, protocol>. After preprocessing the raw flow data, ER-ERT extracts a feature representation from packet payload bytes by the SPA and a feature representation from the flow sequence by the TEM. The two representations, with some stage-based RAT features extracted by the STA, are combined into a feature vector. The feature vector is fed into the fully connected network to identify whether f_i belongs to the network flow of a RAT.

The SPA distills the spatial features of network flow by modeling the local dependency of adjacent packet payloads using the n-gram embedding and autoencoders. Local representations are integrated into an overall picture by Bi-GRU to model the long-term dependency of packets.

The TEM distills the temporal features of a network flow by converting its packet sequences into an image. The RKHS embedding algorithm can convert the network flow sequence into an image to generate a compact and efficient feature representation. DL network CNN further reduces the dimension of the representation.

The STA extracts 11 features from different phases of RAT traffic flow, shown in Table I.

B. Data Preprocessing phase

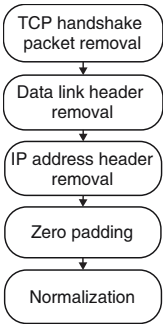


Fig. 2: The data preprocessing process of ER-ERT model.

The data preprocessing of traffic flow has five steps: TCP handshake packet removal, data link header removal, IP address removal, zero padding, and normalization. Fig. 2 shows an overview of the preprocessing procedure.

The TCP handshake packet removal module removes the first three packets in a flow. These packets only contain the TCP handshake information needed for establishing a connection but do not carry any information that can be used to identify the specific application.

The data link header removal module removes the Ethernet header of a packet in the flow. Data link layer information, such as MAC address, type of frame, etc., is useless in packet classification [21]. Removing Ethernet header information can reduce the input size and improve the accuracy of malicious traffic detection.

Keeping the IP address field in a packet as input may cause overfitting of the classification model. The IP address removal module abandons the IP address field in a packet to deal with the overfitting problem. In a real-world emulation, the IP server used by RAT is always a proxy that cannot provide any useful information about the real malicious server.

Each traffic flow contains several time-related packets with different lengths in the interval from 0 to 1500 bytes as most of the computer networks are constrained by Maximum Transmission Unit (MTU) size [7]. We zero-pad the IP packets less than 1500 bytes at the end and truncate the packets longer than 1500 bytes to keep input data length consistent.

TABLE I: FEATURE SET FROM DIFFERENT STAGES OF TRAFFIC FLOW

Stage	Name	Description
Keep-alive Stage	t_α	Proportion of keep-alive stage duration
	p_{h1}	Proportion of keep-alive stage packet number
	p_{h2}	Proportion of keep-alive stage byte number
	h	Whether keep-alive stage exist. if exist, h is 1; otherwise, h is 0
	m_t	Mean packet inter-arrival time
Data Exchange Stage	t_β	Proportion of data exchange stage duration
	p_d	Ratio of upload byte number to download byte number
Early Stage	e_{out}	Upload byte number in early stage
	e_{in}	Download byte number in early stage
	p_{out}	Upload packet number in early stage
	p_{in}	Download packet number in early stage

To handle the packet content more efficiently, we convert the byte in a packet to an integer from 0 to 255 (8 bits). Then, normalize the integer by dividing it by 255 so that all the input values are in the range $[0, 1]$.

C. Feature extraction phase

Algorithm 1: Keep-alive stage segmentation for encrypted traffic flow

Input: An encrypted traffic stream $F(x) = \{p_1, p_2, ..., p_n\}$
Output: Keep-alive time sequence $T(x) = \{(t_1, t_2), ..., (t_n - 1, t_n)\}$

$hp = \{p_1\}$;
$HP = \{\}$;
for $(i = 1; i < n; i++)$ **do**
  if *the arrival time of* p_{i+1} *is latter than the arrival time of* p_i *for more than 1s* **then**
    hp is added to set HP;
    $hp = \{p_{i+1}\}$;
  end
  else
    p_{i+1} is add to hp;
  end
end
Using the K-means algorithm to cluster the set HP according to packet number, average packet size, and the variance of average packet size to get the clustering result $HP_{cluster}$.
Using SVM to classify $HP_{cluster}$ to identify keep-alive packet clusters.
for *all* hp_j *in HP* **do**
  if hp_j *belongs to a keep-alive packet cluster* **then**
    The time interval (t_j, t_{j+1}) between the arrival time of the previous packet of hp_j and the arrival time of the last packet in hp_j is treated as keep-alive time and is added to $T(x)$.
  end
end

a) STA: According to [15], the flow of RAT can be divided into three distinct stages: early stage, keep-alive stage, and data exchange stage. The early stage appears at the beginning of the flow and only lasts for a short time to accomplish basic information exchange. In contrast, the keep-alive and the data exchange stage appear alternately and repeatedly

afterward. The keep-alive stage has the most extended duration in RAT traffic. During the keep-alive stage, the client sends small heartbeat message packets to the server to inform the hacker that it is online. During the data exchange stage, the server sends command requests to the client, and the client returns results to the server. Packets exchanged during this stage are usually very intensive.

We extract different kinds of features during different stages. The utilized feature set is given in Table I. The traffic of normal applications is different from RATs on these stage-based features. For example, the average proportion of the packet number in the keep-alive stage is 15.1% for the traffic of Gh0st RAT and 6.84% for the traffic of normal application. The traffic of Gh0st is collected by ourselves, and the traffic of normal application is from the benign part of the USTC-TFC2016 dataset [22]. Thus, these stage-based features can serve as distinctive characteristics for RAT traffic detection.

Accurately segmenting a session into different stages is an essential prerequisite for extracting these staged features for RAT traffic detection. We design different segmentation algorithms for different stages. The keep-alive stage segmentation algorithm is depicted in Algorithm 1. After dropping the first three handshake packets in the flow, the early stage begins at the fourth packet and ends when the arrival time interval between two adjacent packets is larger than t. The variable t is usually assigned the value one according to experience. Meanwhile, other packets not contained in the early or keep-alive stages are included in the data exchange stage.

b) TEM: The TEM aims to model the flow sequence in network traffic to discover the network behavior characteristics. The framework of this module is depicted in Fig. 3.

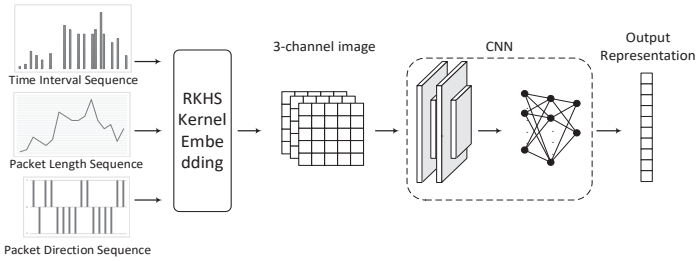


Fig. 3: The architecture of TEM.

Generally, the flow sequence can be modeled as packet inter-arrival time sequence, packet length sequence, and packet direction sequence. So we use these three sequences as the input to the TEM. The detailed description of these sequences is depicted in Table II. To combine these sequences more effectively, we transform them into an image and treat this image as an input to CNN. We use the RKHS embedding to realize this transformation, which is a useful method to represent a (conditional) distribution as an element or an operator in RKHS [14][23]. The theory of RKHS embedding is described below.

The kernel mean embedding κ_x of a marginal distribution P(x) is defined as the expectation of its feature map:

$$\kappa_x = \mathbb{E}[\phi(X)] = \int_x \phi(x)\, dP(x) \tag{1}$$

in which $\phi(x)$ means the high-dimensional mapping of variable x in RKHS through positive definite kernel. Thus, κ_x equals the expectation of $\phi(x)$ with respect to $P(x)$.

To make the input data format of the CNN more like an image, we also define

$$\kappa_x^2 = \mathbb{E}[\phi(X) \times \phi(X)] = \int_x \phi(x) \times \phi(x)\, dP(x) \tag{2}$$

According to the theorem above, packet length sequence s_j, packet inter-arrival time sequence Δt_j, and packet direction sequence d_j can be transformed into the embedding κ_s^2, κ_t^2 and κ_d^2, together constituting a 3-channel image. Then the 3-channel image is fed to the CNN to generate a hidden low-dimensional representation of traffic flow. Table III shows the specification of the CNN used in this paper.

c) SPA: The SPA aims to model the packet payload bytes in network flow. It also explores the impact of packet payload dependencies on identifying malicious RAT traffic. Thus, the n-gram embedding [24][25], and the sliding-window mechanism are involved in generating the feature vector of traffic flow. The autoencoder is used to reduce the dimension of the payload vector. We choose autoencoder because it can model complex data with greater efficiency over shallow machine learning methods and learn latent representation from high-dimensional input data. The procedure of n-gram embedding is shown in Fig. 4. We first use a sliding window of length n to form n-grams (n $\in \{1, 2\}$). Formally, the packet sequence of a flow f is denoted by $f = (p_1, p_2, ..., p_l)$, in which p_i is the payload of the packet i and l is the length of the packet sequence. $p_i = (b_{i1}, b_{i2}, \ldots b_{id})$, where b_{ij} is the j-th byte in payload p_i and d is the length of the packet payload. The sequence of n-grams is denoted by $g = (g_1^n, g_2^{n+1}, ..., g_{l-n+1}^l)$, where g_i^{i+n-1} is the concatenation of vectors from p_i to p_{i+n-1}. Then the autoencoder is trained with the input g_i^{i+n-1}, and it serves as a mapping function $\sigma : g_i^{i+n-1} \Rightarrow \omega_i$. The length of the embedding unit ω_i is 64.

GRU and LSTM can model the long-term dependency of the packet payload, but GRU has fewer parameters than LSTM.

TABLE II: THE INPUT OF TEM

Name	Description
$s_1, s_2, ..., s_n$	The packet length sequence of a bidirectional flow. N is the number of packets in the flow.
$\Delta t_1, \Delta t_2, ..., \Delta t_n$	The packet inter-arrival time sequence of a bidirectional flow.
$d_1, d_2, ..., d_n$	The packet direction sequence of a bidirectional flow, $d_i = 1$ indicates that the i-th packet is sent from server to client.

TABLE III: THE STRUCTURE OF CNN

conv1		conv2		fc1	fc2	output
conv/relu	pool	conv/relu	pool	fc	fc	fc
$3 \times 3 \times 3 \times 32$	max	$3 \times 3 \times 32 \times 64$	max	6400×256	256×64	20

So we choose GRU to extract the feature vector representing the entire flow. Fig. 5 shows the architecture of the Bi-GRU model we use. Two Bi-GRU networks with the same structure are utilized to deal with 1-gram and 2-gram embedding, respectively. Given the embedding sequence $\omega = (\omega_1, ..., \omega_l)$, the Bi-GRU contains a forward GRU network $\overrightarrow{GRU}$ which reads ω from ω_1 to ω_l and a backward GRU network $\overleftarrow{GRU}$ which reads ω from ω_l to ω_1:

$$\overrightarrow{h}_t = \overrightarrow{GRU}(\overrightarrow{h}_{t-1}, w_t), t \in [1, l] \tag{3}$$

$$\overleftarrow{h}_t = \overrightarrow{GRU}(\overleftarrow{h}_{t+1}, w_t), t \in [1, l] \tag{4}$$

where $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ are the forward and backward hidden states, respectively, with a dimension of 16. The initial hidden state vectors $\overrightarrow{h}_0$ and $\overleftarrow{h}_{l+1}$ are both zero vectors.

We stack the 2-layer Bi-GRU in our model, and the two hidden vectors of the first layer form the input vector for the second layer:

$$\overrightarrow{h}^2_t = \overrightarrow{GRU}_2(\overrightarrow{h}^2_{t-1}, (\overrightarrow{h}^1_t, \overleftarrow{h}^1_t)), t \in [1, l] \tag{5}$$

$$\overleftarrow{h}^2_t = \overleftarrow{GRU}_2(\overleftarrow{h}^2_{t+1}, (\overrightarrow{h}^1_t, \overleftarrow{h}^1_t)), t \in [1, l] \tag{6}$$

The output vector o of the Bi-GRU is the concatenation of the final hidden states of both forward and backward directions of all the layers:

$$o = \left[\overrightarrow{h}^1_l, \overleftarrow{h}^1_1, \overrightarrow{h}^2_l, \overleftarrow{h}^2_1\right] \tag{7}$$

The outputs of the two Bi-GRUs with the input 1-gram embedding sequence and 2-gram embedding sequence are then concentrated to form the final representation of the whole flow.

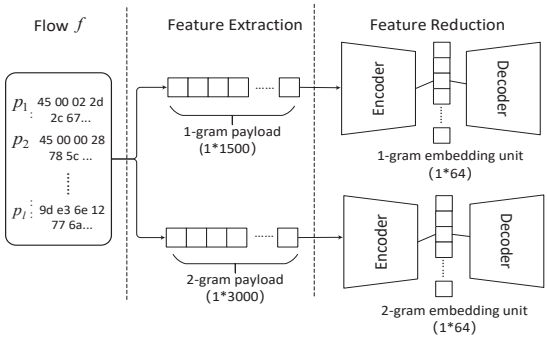


Fig. 4: The n-gram embedding of packet payloads in SPA with autoencoders.

D. Classification phase

The classification phase of our model consists of a fully connected network with one dropout layer and three full connection layers. It takes the representation generated by the Feature Extraction Phase as input and outputs the classification result indicating whether a flow is a RAT flow.

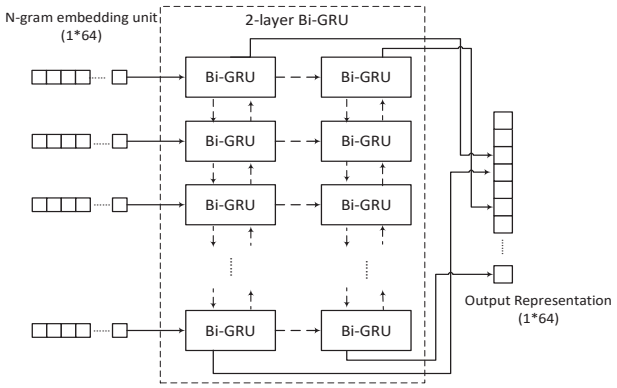


Fig. 5: The embedding of traffic flow in SPA with Bi-GRU.

TABLE IV: AN OVERVIEW OF BENIGN TRAFFIC DATASETS

Name	Session Num	Description
B_1	345470	Containing traffic of BitTorrent, Facetime, FTP, Gmail, MySQL, Outlook, Skype, SMB, Weibo, WorldOfWarcraft
B_2	144840	Collected during 2022.04 on the enterprise network gateway. Containing traffic of WeChat, TeamViewer, Tencent Meeting, etc., which are frequently used by enterprise staff.

IV. EXPERIMENTAL EVALUATION

In this section, we introduce the composition of the dataset and how to evaluate the performance of our model. Then we compare our method with the state-of-the-art methods and analyze the influence of the hyperparameters and their corresponding optimal choice.

A. Data collection

In our experiment, two benign parts of traffic are used for evaluation. The first benign part of the dataset is from the open dataset USTC-TFC2016 [22], and the second benign part of the dataset is collected on a company network port in a realistic environment. The malicious traffic in our dataset is generated by open-source RATs collected by ourselves.

a) Encrypted Benign Traffic Datasets: B_1 is the first benign part of our dataset. In recent studies, USTC-TFC2016 has been widely used for encrypted traffic classification [5]. The benign part of the USTC-TFC2016 dataset organizes 345470 sessions generated from 10 applications. B_2 is the second benign part of our dataset, collected on the gateway server of an enterprise network. We captured all the transport layer security (TLS) encrypted packets on port 443 and segmented the packets into multiple sessions. Then, we removed small sessions which were not sufficiently informative(less than 784 payload bytes). After cleaning the captured traffic data, B_2 organizes 144840 sessions from applications different from the B_1 dataset. The composition of the dataset B_1 and B_2 are detailed in Table IV.

b) Encrypted Malicious Traffic Dataset (M): To hide the real identity of hackers, adopting customized open-source RATs for attacks has been popular in recent years [26]. We select seven open-source RATs based on popularity and stability to generate M. The composition of the dataset M

TABLE V: AN OVERVIEW OF MALICIOUS TRAFFIC DATASETS

RATs	Pupy	Metasploit	QuasarRAT	Gh0stRAT
Session Num	2040	2482	1293	1150
RATs	Remcos	DarkCome	NanoCore	Total
Session Num	1568	1933	1375	11841

is detailed in Table V. We use a client with Windows 10 operating system running on it, serving as a victim, and a RAT server with Ubuntu 16.04 operating system running on it. We collect the traffic between the server and the client to generate dataset M. Five randomly chosen commands (e.g., file downloading, screen shooting, keyboard logging, file uploading, event logging) are executed for each malicious session to simulate the practical usage of RAT samples. The execution interval between two commands is randomly selected in the range of 1 and 60 seconds. The Dataset M contains 11841 different sessions.

We use dataset B_1 and B_2 separately with malicious dataset M to evaluate the efficiency of our model. We use the 5-fold cross-evaluation strategy for persuasive evaluation to acquire a stable performance. The dataset is divided into three parts, training, validation, and testing. The three parts in a fold are divided following the ratio of 0.6:0.2:0.2.

B. Baseline Method and Evaluation Metrics

We choose five baseline methods in this paper as comparisons according to Section 2. The first baseline is a TML-based method proposed by Stergiopoulos et al.[4]. Algorithm CART is used with five side-channel features to classify encrypted traffic. The second baseline method is also a TML-based state-of-the-art algorithm proposed by Gezer at el.[1], which used the Random Forest algorithm to identify TrickBot malware samples. The third baseline method "Deep Packet" is generated by Lotfollahi et al.[7]. The network SAE is used to classify network flow into different applications. The fourth baseline is the model BGRUA proposed by Liu et al.[8]. The model aims to extract forward and backward features of the byte sequences in a session to classify HTTPS traffic. The fifth baseline is the method FS-Net proposed by Liu et al.[11], which used an autoencoder to learn the hidden representation of a traffic flow from the packet length sequence.

The evaluation metrics are precision rate (PR), recall rate (RC), and F1 score. The precision rate indicates how much malicious encrypted traffic labeled by our model is the real RAT traffic. The recall rate indicates how much RAT encrypted traffic is predicted to be malicious. The F1 score considers the precision rate and the recall rate simultaneously to form a compositive metric of the designed model.

C. Overall Effectiveness

In this section, we aim to evaluate the effectiveness of ER-ERT. The specific settings of the experiments are shown in Table VI. Table VII shows the performance of our model on Dataset1, Dataset2, and Dataset3. The result on Dataset1 shows that our model outperforms other models according to the F1 score. The precision of ER-ERT is approximately 97.2%, and the recall rate is 98.0%. Our model outperforms other models on Dataset2 as well, with a detection precision of 96.8% and a recall rate of 97.2%. Compared with the result on Dataset1 and Dataset2, most baseline models suffer a performance loss on Dataset3 in the case of different application types in the training and testing dataset. Nevertheless, our model maintains a high detection precision of 97.0% and a high recall rate of 96.5%, showing the stability of our model.

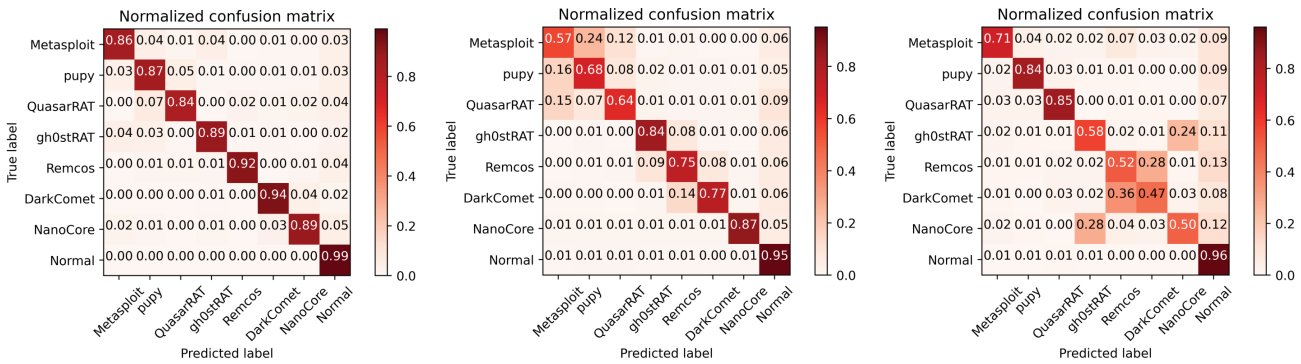
TABLE VI: EXPERIMENT SETTING

	Dataset1			Dataset2			Dataset3		
	train	validation	test	train	validation	test	train	validation	test
Benign	B1	B1	B1	B2	B2	B2	B1	B1	B2
Malicious	M	M	M	M	M	M	M	M	M

TABLE VII: THE RESULT ON DATASET1, DATASET2 AND DATASET3

Dataset	Method	Performance		
		PR	RC	F1
Dataset1	CART [4]	0.702	0.952	0.808
	RF [1]	0.676	0.891	0.769
	Deep Packet [7]	0.843	0.922	0.881
	FS-Net [11]	0.876	0.957	0.914
	BGRUA [8]	0.919	**0.982**	0.949
	ER-ERT	**0.972**	0.980	**0.975**
Dataset2	CART [4]	0.699	0.840	0.763
	RF [1]	0.682	0.857	0.760
	Deep Packet [7]	0.851	0.919	0.884
	FS-Net [11]	0.825	0.908	0.865
	BGRUA [8]	0.917	0.954	0.935
	ER-ERT	**0.968**	**0.972**	**0.970**
Dataset3	CART [4]	0.654	0.728	0.689
	RF [1]	0.590	0.792	0.676
	Deep Packet [7]	0.753	0.886	0.814
	FS-Net [11]	0.833	0.897	0.864
	BGRUA [8]	0.895	0.937	0.916
	ER-ERT	**0.970**	**0.965**	**0.967**

Fig. 6 depicts the confusion matrix corresponding to the classification of 7 kinds of RATs and the benign traffic on Dataset3, showing the per-class ratio of samples correctly and incorrectly classified. The confusion matrix of the model ER-ERT, BGRUA, and FS-Net are depicted in Figs. 6(a) to 6(c), respectively. We compare our model's confusion matrix with the BGRUA's confusion matrix because the BGRUA performs best among the methods that learn representations from packet payload. We compare our model's confusion matrix with the Fs-Net's confusion matrix because the Fs-Net performs best among the methods that learn representations from the flow sequence. We observe that our model performs better in identifying different RAT samples than the BGRUA and FS-Net models. The misclassification of the BGRUA model often occurs when two RAT samples have identical cipher suites, signature algorithms, and other characteristics in the TLS handshake phase, e.g., Pupy and Metasploit. On the other hand, FS-Net has difficulties distinguishing between RATs with similar packet length sequences, e.g., Remcos and DarkComet. Generally, our model outperforms other models

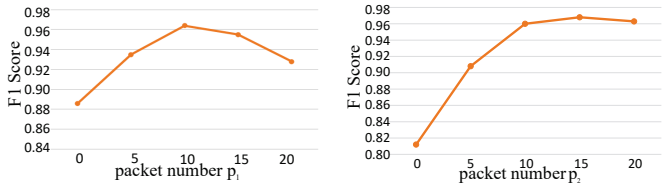


(a) The confusion matrix for model ER-ERT (b) The confusion matrix for model BGRUA (c) The confusion matrix for model FS-Net

Fig. 6: The classification result of different kinds of RATs and normal traffic

in identifying different RATs by simultaneously learning representation from flow sequence and packet payload bytes, with some statistical stage-based attributes.

D. Parameter Tuning



(a) (b)

Fig. 7: (a) packet number p_1 in the SPA and (b) packet number p_2 in the TEM versus F1 score.

We go a step further to experiment on the hyperparameters of ER-ERT. These parameters include packet number p_1 in the SPA and packet number p_2 in the TEM. At the same time, we try to verify the importance of different modules of our integrated model. When we set the parameter p_1 or p_2 to 0, we eliminate the corresponding module. Considering the parameter p_1, the result is shown in Fig. 7(a). If we remove the SPA when p_1 is set to 0, the F1 score drops to 88.6%. Fig.7(b) shows the change in model performance with the variation of p_2 value. If we remove the TEM when p_2 is set to 0, the F1 score drops to 81.2%. Generally, the later a packet arrives in a network flow, the more likely it is a packet of the data exchange stage. The packets of the data exchange stage carry encrypted application data, and their payloads vary a lot due to different operations (e.g., downloading or uploading files with different lengths and content). Thus, involving these packets in the detection model will disturb the identification of malicious RAT traffic. However, using too few packets as the module input leads to a loss of network flow information. Thus, the TEM achieves the best performance when p_2 is set to 15. The SPA, which aims to extract representation from packet payload, appears to be more sensitive to the change of payload content and achieves the best performance when p_1 is set to 10.

TABLE VIII: THE EXPERIMENT RESULT ON DIFFERENT COMPOSITION PATTERNS ON DATASET3

Model Composition	PR	RC	F1
STA	0.651	0.890	0.752
SPA	0.808	0.942	0.870
TEM	0.845	0.949	0.893
STA+SPA	0.832	0.961	0.900
STA+TEM	0.903	0.896	0.900
TEM+SPA	0.932	0.952	0.942
STA+SPA+TEM	**0.970**	**0.965**	**0.967**

To evaluate the effectiveness of three feature extraction modules, we experiment with different permutations of these modules. In Table VIII, we can see the effect of different modules on classification accuracy. The result shows that the combination of three modules outperforms other possible permutations, indicating the efficiency of our design. The combination of the three modules has better performance than the composition of SPA and TME, illustrating the contribution of the stage-based statistical features in detecting RAT traffic as a complement to the temporal and spatial features.

We can also observe from Table VIII that the composition of STA and TEM outperforms the composition of STA and SPA, indicating that the TEM plays a more important role than the SPA in detecting malicious traffic. The effectiveness of the SPA is limited since the SPA can be disturbed more easily by payload content which varies a lot due to different operations. The composition of TEM and SPA outperforms the composition of STA and SPA, indicating that the TEM plays a more important role than the STA. It also illustrates that the features automatically learned from flow have better effects than the predefined statistical features, as the DL network can distill more hidden traffic flow features beyond artificial recognition.

To analyze the impact of the proportion of benign and malicious traffic on classification accuracy, we experiment

TABLE IX: THE EXPERIMENT RESULT ON THE VARIANT PROPORTION OF MALICIOUS AND BENIGN TRAFFIC FLOW

Proportion malicious:benign	Evaluation		
	PR	RC	F1
2:5	0.970	0.965	0.967
1:4	0.968	0.950	0.959
1:8	0.972	0.943	0.957
1:16	0.972	0.868	0.917
1:32	0.980	0.790	0.875

with the model in different scenarios. As shown in Table IX, when the malicious and benign traffic reaches the proportion of 2:5, the model achieves the best performance according to the F1 score. With the decline of the ratio of RAT traffic, the precision of the model rises from 97.0% to 98.0% since it is easier for the model to extract characteristics of benign traffic flow with the increase of normal traffic ratio. However, the recall rate of the model drops sharply from 96.5% to 79%, as it is more difficult for the model to extract characteristics of malicious flow with the decrease of RAT traffic ratio.

CONCLUSION AND FUTURE WORK

In recent years, many DL-based methods have been proposed to address malicious traffic detection problems. These methods generate representations from flow sequences or packet payload bytes. None of these methods simultaneously learn embeddings from flow sequence and packet payload bytes. In this paper, we build an ensemble representation learning model ER-ERT for malicious RAT traffic detection. ER-ERT learns embeddings from flow sequence with CNN and the RKHS embedding and from packet payload bytes with autoencoder and Bi-GRU. The learned representations are combined with some stage-based attributes to enhance the identification ability of RAT traffic behaviors by comprehensively modeling network traffic. According to our experiments, ER-ERT achieves a better performance than state-of-the-art algorithms. Future work will be to modify our model to adapt to the imbalanced training data set.

REFERENCES

[1] Ali Gezer, Gary Warner, Clifford Wilson, and Prakash Shrestha. A flow-based approach for trickbot banking trojan detection. *Computers & Security*, 84:179–192, 2019.

[2] Shadi Aljawarneh, Monther Aldwairi, and Muneer Bani Yassein. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*, 25:152–160, 2018.

[3] Blake Anderson, Subharthi Paul, and David McGrew. Deciphering malware's use of tls (without decryption). *Journal of Computer Virology and Hacking Techniques*, 14:195–211, 2018.

[4] George Stergiopoulos, Alexander Talavari, Evangelos Bitsikas, and Dimitris Gritzalis. Automatic detection of various malicious traffic using side channel features on tcp packets. In *Computer Security: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I 23*, pages 346–362. Springer, 2018.

[5] Cong Dong, Zhigang Lu, Zelin Cui, Baoxu Liu, and Kai Chen. Mbtree: detecting encryption rats communication using malicious behavior tree. *IEEE Transactions on Information Forensics and Security*, 16:3589–3603, 2021.

[6] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. Malware traffic classification using convolutional neural network for representation learning. In *2017 International conference on information networking (ICOIN)*, pages 712–717. IEEE, 2017.

[7] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, 24(3):1999–2012, 2020.

[8] Xun Liu, Junling You, Yulei Wu, Tong Li, Liangxiong Li, Zheyuan Zhang, and Jingguo Ge. Attention-based bidirectional gru networks for efficient https traffic classification. *Information Sciences*, 541:297–315, 2020.

[9] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapé. Distiller: Encrypted traffic classification via multimodal multitask deep learning. *Journal of Network and Computer Applications*, 183:102985, 2021.

[10] Tal Shapira and Yuval Shavitt. Flowpic: Encrypted internet traffic classification is as easy as image recognition. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 680–687. IEEE, 2019.

[11] Chang Liu, Longtao He, Gang Xiong, Zigang Cao, and Zhen Li. Fs-net: A flow sequence network for encrypted traffic classification. In *IEEE INFOCOM 2019-IEEE Conference On Computer Communications*, pages 1171–1179. IEEE, 2019.

[12] Liyan Yang, Yubo Song, Shang Gao, Aiqun Hu, and Bin Xiao. Griffin: Real-time network intrusion detection system via ensemble of autoencoder in sdn. *IEEE Transactions on Network and Service Management*, 19(3):2269–2281, 2022.

[13] Zhuoqun Fu, Mingxuan Liu, Yue Qin, Jia Zhang, Yuan Zou, Qilei Yin, Qi Li, and Haixin Duan. Encrypted malware traffic detection via graph-based network analysis. In *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 495–509, 2022.

[14] Kenji Fukumizu, Le Song, and Arthur Gretton. Kernel bayes' rule: Bayesian inference with positive definite kernels. *The Journal of Machine Learning Research*, 14(1):3753–3783, 2013.

[15] Chun Guo, Zihua Song, Yuan Ping, Guowei Shen, Yuhei Cui, and Chaohui Jiang. Pratd: A phased remote access trojan detection method with double-sided features. *Electronics*, 9(11):1894, 2020.

[16] Michael Finsterbusch, Chris Richter, Eduardo Rocha, Jean-Alexander Muller, and Klaus Hanssgen. A survey of payload-based traffic classification approaches. *IEEE Communications Surveys & Tutorials*, 16(2):1135–1156, 2013.

[17] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of the 13th international conference on World Wide Web*, pages 512–521, 2004.

[18] Soheil Hassas Yeganeh, Milad Eftekhar, Yashar Ganjali, Ram Keralapura, and Antonio Nucci. Cute: Traffic classification using terms. In *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE, 2012.

[19] Pan Wang, Xuejiao Chen, Feng Ye, and Zhixin Sun. A survey of techniques for mobile service encrypted traffic classification using deep learning. *IEEE Access*, 7:54024–54033, 2019.

[20] Julian Busch, Anton Kocheturov, Volker Tresp, and Thomas Seidl. Nf-gnn: Network flow graph neural networks for malware detection and classification. In *33rd International Conference on Scientific and Statistical Database Management*, pages 121–132, 2021.

[21] Pan Wang, Feng Ye, Xuejiao Chen, and Yi Qian. Datanet: Deep learning based encrypted network traffic classification in sdn home gateway. *IEEE Access*, 6:55380–55391, 2018.

[22] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. A survey of network-based intrusion detection data sets. *Computers & Security*, 86:147–167, 2019.

[23] Le Song, Kenji Fukumizu, and Arthur Gretton. Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, 30(4):98–111, 2013.

[24] Marc Damashek. Gauging similarity with n-grams: Language-independent categorization of text. *Science*, 267(5199):843–848, 1995.

[25] Yafei Sang, Yongzheng Zhang, and Chengwei Peng. Pronet: Toward payload-driven protocol fingerprinting via convolutions and embeddings. In *Collaborative Computing: Networking, Applications and Worksharing: 13th International Conference, CollaborateCom 2017, Edinburgh,*

UK, December 11–13, 2017, Proceedings 13, pages 519–529. Springer, 2018.

[26] Y. Ishikawa. Open source as fuel of recent apt. https://hitcon.org/2017/pacific/0composition/pdf/Day1/R1/R1-4.12.7.pdf. Accessed Jan.24 2022.