IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 26, 2020, accepted August 20, 2020, date of publication October 12, 2020, date of current version November 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3023434

A Worm Detection System Based on Deep Learning

HANXUN ZHOU[1], YESHUAI HU[1], XINLIN YANG[1], HONG PAN[2], WEI GUO[3], AND CLIFF C. ZOU[4], (Senior Member, IEEE)

[1]School of Information, Liaoning University, Shenyang 110036, China
[2]School of Economics, Liaoning University, Shenyang 110036, China
[3]School of Computer, Shenyang Aerospace University, Shenyang 110136, China
[4]Department of Computer Science, University of Central Florida, Orlando, FL 32816, USA

Corresponding authors: Hong Pan (panhong@lnu.edu.cn) and Wei Guo (guowei@sau.edu.cn)

This work was supported in part by the National Key Research and Development Program under Grant 2019YFB1406002, in part by the National Science Foundation of China under Grant 51704138, in part by the Key Scientific Research Project of Liaoning Provincial Department of Education under Grant LZD202002, in part by the Liaoning Education Department under Grant JYT19053, in part by the National Natural Science Foundation of Liaoning under Grant 2020-MS-239, and in part by the U.S. National Science Foundation under Grant DGE-1915780.

ABSTRACT In today's cyber world, worms pose a great threat to the global network infrastructure. In this paper, we propose a worm detection system based on deep learning. It includes two main modules: one worm detection module based on a convolutional neural network (CNN) and one automatic worm signature generation module based on a deep neural network (DNN). In the CNN-based worm detection module, we propose three kinds of data preprocessing methods: frequency processing, frequency weighted processing, and difference processing, and use CNN to train the model for worm detection. In the DNN-based worm signature generation module, there are two phrase: DNN is firstly utilized for training the model with worm payloads and their corresponding signatures as input in the training phrase. After worm payloads are fed into the trained DNN model in the test phrase, worm signatures are generated by our proposed Signature Beam Search algorithm. In the experiment, we firstly analyzed the impact of different data preprocessing methods and the number of convolution-pooling layers in the CNN model on the worm detection performance. Then we analyzed the effects of different signatures in the DNN algorithm on the automatic generation of worm signatures. Experiments show that the generated signatures have a good detection performance.

INDEX TERMS Network security, worm detection, worm signature automatic generation, deep learning.

I. INTRODUCTION

Cyber threats from Internet worms are not new, but how to effectively detect and defend against them still remains an ongoing challenge. For example, global financial and economic losses from the "WannaCry" attack that crippled computers in at least 150 countries could swell into the billions of dollars, making it one of the most damaging incidents involving the so-called ransomware [1].

One of the most common and effective ways to detect worm attacks is to implement a signature-based intrusion detection system (IDS). However, signatures are usually analyzed and generated manually by security experts after worms have already launched attacks and caused severe damage. Furthermore, it is easy to fool signature-based solutions by obfuscation [2]. This technique simply skirts around the signature database stored in the IDS, giving the hacker an ideal opportunity to gain access to the network. As a result, the key to worm signature generation is to find matching invariant strings from worm payloads as soon as possible.

In recent years, there are a large number of researches on the automatic generation of worm signatures. Kaur and Singh [2] provided a detailed survey to outline the research efforts to the detection of modern zero-day malware in the form of zero-day polymorphic worms. Aljawarneh *et al.* [3] investigated the current automatic methods used to generate efficient and accurate signatures to create countermeasures against attacks by polymorphic worms. Bayoğlu *et al.* [4] proposed a new polymorphic worm signature scheme called Conjunction of Combinational Motifs (CCM). CCM utilizes

The associate editor coordinating the review of this manuscript and approving it for publication was Pengcheng Liu.

common substrings of polymorphic worm copies and also the relation between those substrings through dependency analysis. Tang *et al.* [5] designed a network-based signature generation (NSG) system—PolyTree, to defend against polymorphic worms. They observed that signatures from worms and their variants are relevant and a tree structure can accurately reflect their familiar resemblance. Mondal *et al.* [6] declared an automatic method that will generate signatures for the detection of polymorphic worms, and they applied Principal Component Analysis (PCA) for determining the critical substrings that appear mostly and are pooled amongst the instances of polymorphic worms for using them as signatures. Eskandari *et al.* [7] proposed a signature generation scheme based on token extraction and multiple sequence alignment. However, these methods are usually based on the features manually presented, and then design an algorithm to detect worms. As a result, the performance is much more dependent on those manually pre-defined features which may become the bottleneck.

Nowadays, significant achievements have been achieved in the fields of image processing, video processing, and natural language because of the remarkable technological breakthrough in feature learning in deep learning [8], [9]. Researchers feed the computer a learning algorithm, expose it to data to train it, and then allow the computer to figure out by itself how to recognize the desired objects, words, sentences, or speeches. Therefore, the cybersecurity academic community has begun to pay attention to the application of deep learning in intrusion detection. Zhu *et al.* [10] introduced DeepFlow, a novel deep learning-based approach for identifying malware directly from the data flows in the Android application. Özkan *et al.* [11] contributed the CNN features to overcome the malware detection problem. Kim *et al.* [12] proposed a convolutional gated recurrent neural network model that is capable of classifying malware to their respective families. Although there are already some researches on malware detection that utilizes deep learning to detect malicious code, most of them are classification. They can not help security products, for example virus products, detect attacks.

Worms spread over networks by payloads to exploit vulnerabilities in operating system or installed software. Payloads usually perform actions on affected computers. Therefore, payloads can distinguish worms. Due to the capacity of deep learning on learning features automatically, we choose to use deep learning to detect worms and extract worm signatures.

In this paper, we propose a novel worm detection system based on deep learning, which can detect worms accurately and generate worm signatures automatically. It includes two core parts: one CNN-based worm detection module and one DNN-based worm signature generation module. In the CNN-based worm detection module, three payload preprocessing methods are proposed: frequency processing, frequency weighted processing, and difference processing. CNN is utilized for training the model with processed payloads as inputs. In the DNN-based worm signature generation module, DNN is used to learn from worm payloads and their corresponding signatures to obtain the DNN model. Then we present a new signature generation algorithm called Signature Beam Search. Worm payloads are fed into the trained DNN model, and the Signature Beam Search algorithm is used to generate worm signatures. In the experiments, we firstly analyzed the effects of different preprocessing methods and the number of convolution-pooling layers used by CNN on detection. Then we analyzed the performance of DNN to extract signatures. Experiments show that the generated signatures have both low false positives and low false negatives.

The rest of the paper is organized as follows: we discuss the related work of worm detection in Section II. We introduce the system architecture of the worm detection system based on deep learning in Section III. Section IV introduces the worm detection approach based on CNN. The DNN-based automatic worm signature generation approach is described in Section V. Section VI discusses our extensive experiments in evaluating the proposed worm detection system. In the end, section VII draws the conclusion.

II. RELATED WORK

There are two types of intrusion detection systems: anomaly-based and signature-based intrusion detection systems. Anomaly-based intrusion detection system regularly monitors events and compares them with the statistical model. The signature-based detection method used by intrusion prevention systems involves a dictionary of uniquely identifiable signatures located in the code of each exploit. A vital advantage of this method is that signatures are easy to develop and understand for security experts. However, signatures are usually generated after worms have caused damage, so there are researches on automatic generation methods for worm signatures.

References [3], [13] investigated the current automatic methods used to generate efficient and accurate signatures to create countermeasures against attacks by polymorphic worms. Nahmias *et al.* [14] presented TrustSign, a novel, trusted automatic malware signature generation method based on high-level deep features transferred from a VGG-19 neural network model pre-trained on the ImageNet dataset. Szynkiewicz *et al.* [15] developed an efficient algorithm for token extraction and a novel method for automatic multi-token signature composition. Wang *et al.* [16] proposed an automatic signature extraction algorithm for polymorphic worms based on the improved Term Frequency-Inverse Document Frequency (TF-IDF). Afek *et al.* [17] presented a basic tool for zero-day attack signature extraction.

Due to the capacity of learning features automatically by deep learning, researchers apply deep learning to intrusion detection to resolve the characteristic dependence problem mentioned above. Nguyen *et al.* [18] built an IDS platform based on a convolutional neural network (CNN) called IDS-CNN to detect DDoS attacks. Experiments have shown that the CNN-based DDoS detection method outperforms traditional methods such as KNN, SVM, and Naive Bayes. Vinayakumar *et al.* [19] modeled network traffic as

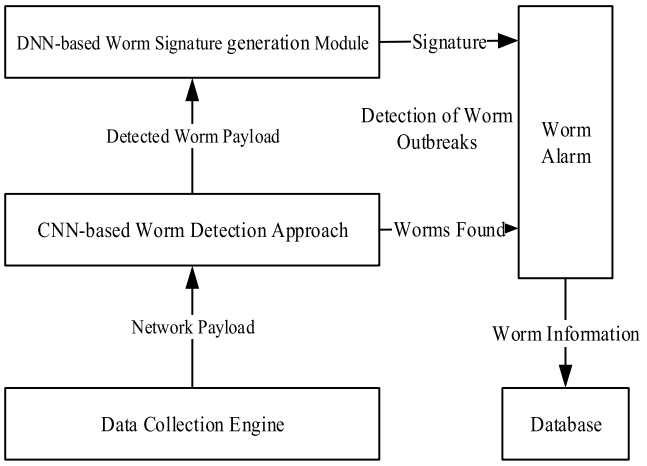


FIGURE 1. System architecture.

time-series, in a predefined time range with supervised learning methods.

Researchers have presented some DNN-based approaches to detect malware that have shown better results. Venkatraman *et al.* [20] proposed a novel and unified hybrid deep learning and visualization approach for the effective detection of malware. Zhong *et al.* [21] proposed a Multi-Level Deep Learning System (MLDLS) that organizes multiple deep learning models using the tree structure.

III. THE SYSTEM ARCHITECTURE

The architecture of our proposed worm detection system based on deep learning is shown in Figure 1. It is mainly divided into four modules, including a data collection engine, CNN-based worm detection module, DNN-based worm signature automatic generation module, and a worm alarm module.

The data collection engine is responsible for collecting network traffic and extracting payloads. The CNN-based worm detection approach detects worms by the payloads processed by the data collection engine. It distinguishes worm payloads from normal payloads based on CNN. If it detects worm, the corresponding payloads are submitted to the DNN-based worm signature generation module, which is responsible for generating signatures automatically. The DNN-based worm signature generation module is trained by learning the characteristics of worm payloads based on historic known worm traffic. Worm payloads are fed into the trained DNN network to generate corresponding signatures. All the results from the three components above are submitted to the worm alarm component, which helps to manage the system. Since the data collection engine and worm alarm are the auxiliary components of the system, this paper will not describe them in detail, but elaborate on the two core components of the CNN-based worm detection method and the DNN-based worm feature generation module.

The formal definitions of the detection process can be described as follows:

TABLE 1. Notations.

Notation	Definition
p_i	The character in the payload in index i
q_i	The ASCII value of p_i
J	j represents the ASCII value of each character
data[j]	The data after preprocessing
$\boldsymbol{A}$	The worm payload
$\boldsymbol{B}$	The signature of the worm payload
M	The length of worm payload
N	The length of the signature
b_c	The context of the signature with a window size of C
X	The set of all worm payloads
Y	The set of all possible signatures
D	The size of the character embedding
V	The size of the vocabulary composed of the worm payload
H	The number of neurons in the hidden layer
Q	The size of the feature extraction window
$[x, y]$	The range of the feature extraction window value
K	The size of the beam

Definition 1: $pl = \{pl_1, \ldots, pl_i, \ldots, pl_M\}$, $i \in (1, M)$ represents a payload, each pl_i represents one byte, M is the length of the payload;

Definition 2: The payload set PL = {pl | pl is the payload collected by the data collection engine}

Definition 3: The CNN payload set CPL = {cpl | cpl $\in$cnn(pl)Λ pl $\in$ PL}, where cnn is the function which represents worm payloads detected by the DNN-based worm signature generation module.

Definition 4: The signature set SIG = $\{\text{sig}_{\text{cpl}} | \ \text{sig}_{\text{cpl}} \in \text{dnn(cpl)}\Lambda\text{cpl} \in \text{CPL}\}$, where dnn() is the function that represents signatures generated by DNN-based worm signature automatic generation module, and sig_{cpl} represents the generated signature of the payload cpl.

The description of the notations used in this paper is given in Table 1.

IV. CNN-BASED WORM DETECTION

With the in-depth research of CNN, it has achieved outstanding classification performance in image recognition. To capture the critical characteristics of worm payloads, we choose to use CNN for worm detection. The variable length of payloads should be first converted to a fixed length because CNN can only deal with fixed-length data. Although vectors, two-dimensional and three-dimensional matrices can all be considered as input to CNN, vectors have the following advantages: they can decrease the storage space and reduce the calculation during training. Therefore, we convert payloads into vectors as the input of CNN.

Definition 5: prep = preprocess(pl), where pl is the payload in Definition 1 and preprocess(pl) is the preprocessing

function which coverts the payload *pl* to the generated vector *prep*.

We use the following three methods to process payloads: frequency processing, frequency weighted processing, and differential processing.

Frequency Processing: payloads are processed by counting the number of occurrences of each byte in the payload. Algorithm 1 illustrates the detailed procedure.

Frequency Weighted Processing: both the byte value and its frequency are considered in this method. Firstly, the frequency of each byte in the payload is counted, and then the weighted operation is performed by multiplying the byte value with its frequency. Algorithm 2 is the specific processing procedure.

Algorithm 1 Frequency Processing

Input: **q**
Output: data[], the data after preprocessing
for $i = 1, \ldots M$ **do**
    data $[q[i]]$ = data $[q[i]] + 1$
End
for $j = 0, \ldots 255$ **do**
    data $[j]$ = (data $[j]$) / n
End

Algorithm 2 Frequency Weighted Processing

Input: **q**
Output: data[], the data after preprocessing
for $i = 1, \ldots M$ **do**
    data $[q[i]]$ = data $[q[i]] + 1$
End
for $j = 0, \ldots 255$ **do**
    data $[j]$ = (data $[j] * j$) / n
End

Algorithm 3 Difference Processing

Input: **q**
Output: data[], the data after preprocessing
for $i = 1, \ldots M - 1$ **do**
    data $[q[i]] += q[i+1] - q[i]$
End
for $j = 1, \ldots$ n **do**
    data $[j]$ = (data $[j]$) / n
End

Difference Processing: based on the 2-gram model, two adjacent bytes in worm payloads are closely correlated. As a result, we process payloads by calculating the difference between two adjacent bytes in the payload. The detail is in Algorithm 3.

Frequency processing represents the byte distribution of payloads. Frequency weighted processing represents the combination of the value and its frequency. Difference processing exploits the relationship of two adjacent bytes. All the methods above can reflect the features of payloads from different perspectives. And they can convert payloads to the corresponding format. We input the corresponding data to CNN to detect worm attacking. Although we tried to utilize other approaches to process payloads, they did not work after the processed data was input to the CNN network.

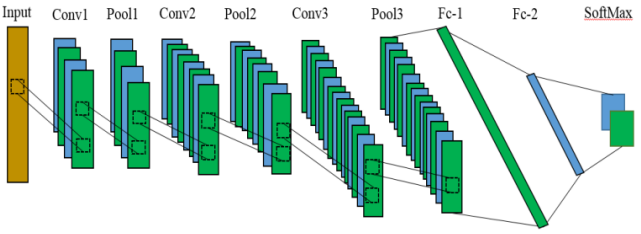


FIGURE 2. CNN model.

We established multilayer convolution-pool CNN models to study the effectiveness of the CNN-based worm detection. In this section, we introduce a three-layer convolution-pooling CNN model as an example, which is shown in Figure 2. The input in the figure represents the preprocessing vector, and Conv-$*$ represents the convolutional layer, Pool-$*$ represents the pooled layer, and Fc-$*$ represents the fully connected layer. The processed payloads are fed into the CNN network as input for training. The convolution layer extracts features through a convolutional operation. The pooling layer down-samples the extracted feature maps. The fully connected layer combines the extracted features into a vector, and the SoftMax layer classifies and outputs the category.

The computational complexity of CNN is o($\sum_{l=1}^{D} M_l^2 K_l^2 C_{l-1} C_l$), where D is the number of layers in CNN, ~~and~~ l is the l-th convolutional layer, ~~and~~ M is the size of feature maps, K is the size of convolutional kernels, and C_l is the output channels of l-th convolutional layer.

V. DNN-BASED WORM SIGNATURE GENERATION

The automatic signature generation helps shorten the response time for identifying malware by dynamically extracting signatures of unknown worms without human intervention. With the development of deep learning, Deep Neural Network (DNN) models have yielded many state-of-the-art results in natural language processing. We choose to use DNN to extract worm signatures automatically, because worm signature generation is also a sequence-to-sequence task, and both payloads and signatures can be considered as a special text.

The whole process for automatic generation of worm signatures is divided into two parts: training the DNN model and generating worm signatures. In the training phase, both worm payloads and their corresponding signatures are fed to the DNN network for training. At the stage of generating signatures, worm payloads are input into the DNN network model, and our newly proposed Signature Beam Search algorithm is used to generate their corresponding signatures.

A. MODEL

Definition 6: The input $\boldsymbol{a}$ is a sequence of M bytes $(a_1, a_2, \ldots, a_M)$ and represents a worm payload.

Definition 7: A sequence $\boldsymbol{b}$ $(b_1, b_2, \ldots, b_N)$ represents a signature, where $b_i (1 < i <= N)$ is also one byte.

Definition 8: X represents the set of all worm payloads, Y represents the set of all possible signatures.

The signature generation algorithm takes worm payloads as input and outputs its corresponding signature. There is a scoring function s: $X \times Y \rightarrow \mathbb{R}$ and the algorithm aims to find signatures $\boldsymbol{b'} \in Y$ so that:

$$b' = \arg\max_{b \in Y} s(a, b) \tag{1}$$

When the score function takes the window into account, it can be approximately represented as:

$$s(a, b) \approx \sum_{i=0}^{N-1} dnn\,(b_{i+1}, a, b_c) \tag{2}$$

where $b_c \triangleq \boldsymbol{b}[i\text{-}c+1, \ldots, i]$ represents the signature context with window C, and function dnn() represents the prediction function. The scoring function s($\boldsymbol{a}$, $\boldsymbol{b}$) can be expressed as the form of conditional log-probability: s($\boldsymbol{a}$, $\boldsymbol{b}$) = log p($\boldsymbol{b}|\boldsymbol{a}$; θ) $\approx$ $\sum$p($\boldsymbol{b}|\boldsymbol{a}$, b_c; θ), where i refers to the index of position in signatures. We make a Markov hypothesis on b_c and suppose that when i <1, b_i is a start symbol <s>. From the above scoring function, we need to model the probability distribution of the local condition: p($b_{i+1}|\boldsymbol{a}, b_c; \theta$). We utilize machine translation to parameterize the conditional probability distribution into a neural network, including a DNN model and a conditional signature generation encoder.

1) DNN MODEL

We construct a deep neural network model with four hidden layers based on the standard feed-forward neural network language model (NNLM) [22]. There are two reasons why we choose to use rectified linear unit(ReLU), for the hidden layer activation function. Firstly, it is more computationally efficient to compute than Sigmoid like functions, because ReLU does not perform expensive exponential operations as in Sigmoid. Secondly, it may reduce the likelihood of the gradient to vanish. The DNN model is as follows:

$$p\,(b_{i+1} \mid b_c, a; \theta) \propto \exp\left(\mathbf{V}h'''' + \mathbf{W} * \mathrm{enc}\,(a, b_c)\right) \tag{3}$$

$$\tilde{b}_c = [\mathbf{E}b_{i-C+1}, \cdots, \mathbf{E}b_i] \tag{4}$$

$$h' = relu\left(\mathbf{U}\tilde{b}_c\right) \tag{5}$$

$$h'' = relu\left(\mathbf{U}'h'\right) \tag{6}$$

$$h''' = relu\left(\mathbf{U}''h''\right) \tag{7}$$

$$h'''' = relu\left(\mathbf{U}'''h'''\right) \tag{8}$$

where enc() is an attention-based encoder that returns a vector of size H, which represents the context of worm payloads and corresponding signatures. The parameters are θ = (**E**, **U**, **U**', **U**'', **U**''', **V**, **W**) where $\mathbf{E} \in \mathrm{R}^{D \times V}$ represents an embedding matrix for signatures, $\mathbf{U} \in \mathbb{R}^{(CD) \times H}$, $\mathbf{U}', \mathbf{U}'', \mathbf{U}''' \in \mathbb{R}^{H \times H}$, $\mathbf{V} \in \mathbb{R}^{V \times H}$, $\mathbf{W} \in \mathbb{R}^{V \times H}$ are weight matrices. D represents the size of the byte embedding, V is the size of the vocabulary composed of worm payloads, H represents the number of neurons in the hidden layer, and C represents the context size in signatures. Figure 3 gives a schematic diagram of a deep neural network model, where $\boldsymbol{a}$ represents worm payloads, and $\boldsymbol{b}$ represents corresponding signatures. Four hidden layers are selected because the performance of the model with fewer layers is worse and the performance of the model with more layers is almost the same. Therefore, the computational complexity is o(|E| + |U| + |U'| + |U''| + |U'''| + |V| + |W|), where |P| is the size of P.

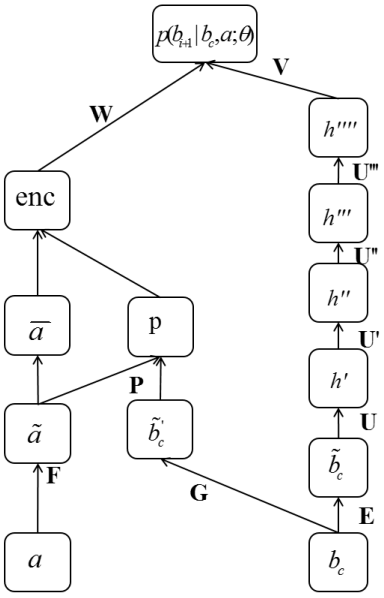


FIGURE 3. DNN model.

2) ATTENTION-BASED ENCODER

Signatures, which may be composed of multiple consecutive sub-strings, are the key information to identify worms. As a result, we only need to focus on the context of a limited number of bytes instead of the entire text. Therefore, we use an Attention-based encoder [23] that can construct representations based on worm payloads and signatures. Given a worm payload, there is a sliding window of size Q which is moving from the very left of the payload to the very right in the encoder. Each time the sliding window moves right by one byte. The specific formulas are described as follows:

$$\mathrm{enc}\,(a, b_c) = \mathrm{p}^{\mathrm{T}}\bar{a} \tag{9}$$

$$\mathrm{p} \propto \exp\left(\tilde{a}\mathbf{P}\tilde{b}_c'\right) \tag{10}$$

$$\tilde{a} = [\mathbf{F}a_1, \ldots, \mathbf{F}a_M] \tag{11}$$

$$\tilde{b}_c' = [\mathbf{G}b_{i-C+1}, \ldots, \mathbf{G}b_i] \tag{12}$$

$$\forall i\bar{a}_i = \sum_{q=i-Q}^{i+Q} (\tilde{a}_i/Q) \times \frac{1}{M} \tag{13}$$

In formulas (9-13), $\mathbf{G}$ *in* $\mathbb{R}^{D \times V}$ represents an embedding of the signature context, $\mathbf{P} \in \mathbb{R}^{H \times (CD)}$ is a weight matrix,

$\mathbf{F} \in \mathbb{R}^{H \times V}$ represents the embedding matrix of payloads; Q is the size of a smoothing window and configured as ten optimized by the data driven method. a is the input(payloads) where a = [$a_1, a_2, \ldots, a_M$]; $\tilde{a}$ can be computed by equation (11), and $\tilde{a}_i$ is the element of $\tilde{a}$; $\bar{a}$ can be calculated by equation (13) and $\bar{a}_i$ is the element of $\bar{a}$; p^{T} is the transpose of p. Therefore, the computational complexity is o(|F| + |G|+2|P| + $|\tilde{a}_i|$-Q), where |U| is the size of U.

We define the model above as $p(y_{i+1}|x, y_c; \theta)$, and then the loss function can be defined as follows:

$$NLL(\theta) = -\sum_{j=1}^{J} \log p(y^{(j)}|x^{(j)}; \theta),$$
$$= -\sum_{j=1}^{J}\sum_{i=1}^{N-1} \log p(y_{i+1}^{(j)}|x^{(j)}, y_c; \theta). \quad (14)$$

where $\mathrm{x}^{(\mathrm{j})}$ is the input (payloads), and $\mathrm{y}^{(\mathrm{j})}$ is the corresponding signatures.

B. WORM SIGNATURE GENERATION

The signature generation algorithm needs to find an optimal signature ***b'*** $\in Y$ as follows:

$$b' = \arg\max_{b \in Y} \sum\nolimits_{i=0}^{N-1} dnn(b_{i+1}, a, b_c) \quad (15)$$

Machine translation is an NP problem, but the computation cost is not large when generating worm signatures. The dictionary V is composed of 256 ASCII in the worm signature generation. Furthermore, the location of the signature in the worm payload is orderly and sequential. Using the above characteristics of signatures, we propose a new method—-Signature Beam Search algorithm to solve argmax function when generating signatures.

The Signature Beam Search algorithm uses a global search to generate worm signatures. In the search process, the appropriate bytes are selected by judging whether the predicted bytes are adjacent to the previously predicted bytes in worm payloads. When generating worm signatures, we select K bytes to the previously predicted bytes for each position in signatures. Before outputting the predicted signatures, the best signature is selected by sorting the log-probability of the candidate K signatures. As a result, the maximum time complexity of the Signature Beam Search algorithm is O(KNV). Signature Beam Search algorithm is described in Algorithm 4.

Algorithm 4 Signature Beam Search Algorithm

Input: $\boldsymbol{a}$, K; a represents the worm payload, K represents the size of beam
Output: sig; sig represents the $N \times K$ dimensional signature
Initialize: sig[0][K] = {<s>}
for $i = 1, \ldots N$ **do**
  for $k = 1, \ldots K$ **do**
    $b_{\text{predict}} = \{b_1, b_2, \ldots, b_V\} = dnn(b_{i+1}, \boldsymbol{a}, b_c)$
    for $v = 1, \ldots V$ **do**
      if b_i1_index == b_i_index +1 **then**
        sig[i][k]= b_{i+1}
        break
      Else
        continue
      End
    end for
    if $v == V$ **then**
      sig[i][k] = 'u'
    End
  end for
end for
return sig

VI. EXPERIMENTS

In our experiments, we implemented both CNN and DNN algorithms for worm detection and signature generation based on the Torch framework [24]. Since most related and well-known methods (like Polygraph and PolyTree) evaluate their performance using synthetically generated payloads which are based on real-world exploits, we follow the same evaluation approach.

We used three synthetic worm payload datasets which are presented by Polygraph: Apache-Knacker [25], ATPhttpd [26], and TSIG [27]. Each payload dataset contains about 5000 records. All synthetically created worms use either the HTTP protocol or DNS protocol, so we collected real-world normal traffic data under the HTTP and DNS protocols. First of all, we collected a 10-day HTTP trace from our campus network gateway. Secondly, we utilized a 5-day DNS trace from a DNS server that served in a company. Finally, the worm payload datasets were randomly combined with the real-world traffic traces.

We randomly divided 80% of the dataset as the training dataset and 20% as the testing dataset in the experiment. For the learning phase, we used a k-fold cross-validation method: the training dataset was partitioned into k subsets randomly. A single subset was retained as the validation dataset for testing the model, while the remaining k-1 subsets of the original dataset were used as training data. We repeated the process for $k = 10$ times, which was the same as [29], [30]; each one of the k subsets had been used once as the validation dataset. To obtain a single estimate, we computed the average of the k results from the folds.

A. CNN-BASED WORM DETECTION

To explore the validation of the CNN model to detect worms, we performed the CNN algorithm by binary classification. Binary classification is to distinguish worm attacks from normal traffic. There are additional experiments, including known and unknown worm detection. For known worm detection, there are both worm and normal payloads except Code Red II worm payloads in the training and test dataset. For the unknown worm detection, the test dataset is composed of all five CodeRed II worm payloads and a private worm

TABLE 2. Accuracy of the test set in binary classification.

Preprocessing Method	Accuracy	FNR	FPR	Precision	Recall	F1-score	AUC	loss
Frequency	98.72%	0.87%	1.02%	92.17%	91.57%	92.01%	97.82%	0.32
Frequency Weight	99.01%	0.63%	0.81%	92.34%	92.01%	92.12%	98.12%	0.23
Difference	99.27%	0.54%	0.79%	92.73%	92.51%	92.37%	98.21%	0.19

payload dataset, and the training dataset consists of both worm payloads which do not contain payloads in the test dataset and normal payloads.

We use accuracy, false positive rate(FPR), false negative rate (FNR), Precision, Recall, F1-score, and AUC to evaluate CNN model as follows:

Accuracy is the fraction of predictions that our model is right. It is defined as follows:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{16}$$

where TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative.

FPR is computed as the ratio between the number of negative events incorrectly classified as positive and the total number of actual negative events:

$$\text{FPR} = \text{FP}/(\text{FP} + \text{TN}) \tag{17}$$

FNR is the proportion of positives that yield negative prediction:

$$\text{FNR} = \text{FN}/(\text{FN} + \text{TP}) \tag{18}$$

Precision is defined as (19), which is the fraction of relevant samples between the retrieved samples:

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP}) \tag{19}$$

The Recall is the fraction of positive samples that are correctly classified as 'positive':

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN}) \tag{20}$$

The harmonic mean of precision and recall defines as F1-score:

$$\text{F1} - \text{score} = 2 * \text{TP}/(2 * \text{TP} + \text{FP} + \text{FN}) \tag{21}$$

Area Under Curve (AUC) measures the trade-off between misclassification rate and FPR:

$$\text{AUC} = 0.5 * (\text{TP}/(\text{TP} + \text{FP}) + \text{TN}/(\text{TN} + \text{FP})) \tag{22}$$

The loss function is defined as follows:

$$loss = -\sum_{j=1}^{n} Y_j \log y_j \tag{23}$$

where n denotes the number of output; Y denotes the output.

1) DATA PREPROCESSING METHODS

In this section, we trained CNN models with data processed by three different methods proposed in Section IV to assess the performance under the condition of one convolution-pooling layer.

The results for binary classification are shown in Table 2. The accuracy of frequency, frequency weight, and difference processing methods are 98.72%, 99.01%, and 99.27%, respectively. The accuracy of the difference processing method is the best. It also outperforms the other two processing methods in other metrics. Therefore, the difference processing approach outperforms the other approaches in binary classification. Figure 4 shows the accuracy trend of three processing methods in binary classification as execution time progresses for the 10-fold cross-validation on the dataset. Considering the accuracy of training data and validation data in binary classification, we observe that the accuracy is steady after the epoch is larger than 6. There are similar results in other metrics, so we do not report the corresponding curve.

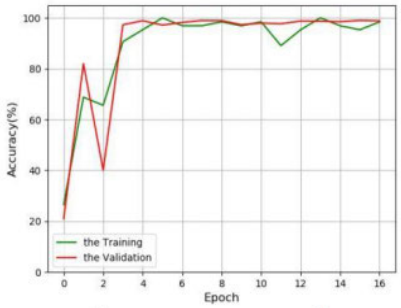
2) CONVOLUTIONAL-POOLING LAYERS

We configured the CNN model with one, two, and three convolution-pooling layers to evaluate the effect of the number of layers on the accuracy of the CNN model. We choose the frequency processing method as an example while others hold similar results.

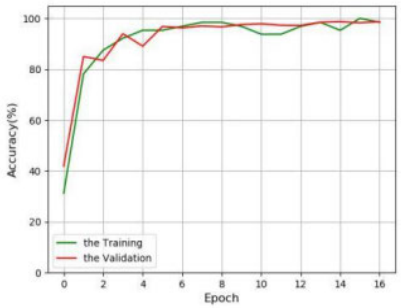
The results achieved by 10-fold cross-validation criterion in binary classification are presented in Table 3 to expound the performance of different layers, and also reveal the general information of the relation between the number of layers and the prediction accuracy for worm detection. The accuracy of one, two, and three layers are 98.72%, 99.07%, and 99.25%, respectively. Generally speaking, the number of convolution-pooling layers has little effect on the accuracy, because the accuracy is only improved by 0.5%. There are similar results to other metrics. Although more layers might improve the accuracy further, we do not report the result since it becomes trivial to simply add more layers, and the improved performance might not compensate for the increased computational cost. However, the less the layers are, the less the parameters are. As a result, one convolution-pooling layer is a good choice due to less occupation of system resources.

B. WORM SIGNATURE GENERATION

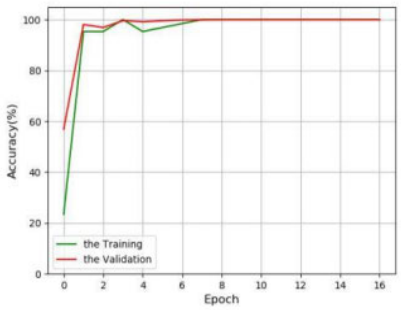
In this experiment, we used signatures generated by Polygraph [11], CCM [4], PolyTree [5] which are shown in Table 4, Table 5, and Table 6. Perplexity is utilized to study



(a) Frequency Processing



(b) Frequency Weight Processing



(c) Difference Processing

FIGURE 4. Accuracy of binary classification.

TABLE 3. Accuracy of known worm detection with the frequency processing method in binary classification based on different layers.

Convolution-pooling layers	Accuracy	FNR	FPR	Precision	Recall	F1-score	AUC	loss
CNN(1 layer)	98.72%	0.87%	1.02%	92.17%	91.57%	92.01%	97.82%	0.32
CNN(2 layer)	99.07%	0.65%	0.83%	92.20%	91.77%	92.10%	98.76%	0.21
CNN(3 layer)	99.25%	0.50%	0.73%	92.87%	92.07%	92.42%	98.91%	0.18

the performance of the DNN model. The formula is as follows:

$$\begin{aligned} \text{Perplexity}(S) &= P(w_1 w_2 \cdots w_N)^{\frac{1}{N}} \\ &= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{p(w_i \mid w_1 w_2 \cdots w_{i-1})}} \end{aligned} \tag{24}$$

TABLE 4. Signature generated by polygraph.

Worm	Signature
Apache-Knacker	GET HTTP/1.1\r\n : \r\nHost: \r\n : \r\nHost: \xFF\xBF \r\n
ATPhttpd	GET \x9e\xf8 HTTP/1.1\r\n
TSIG	\xFF\xBF \x00\x00\xFA

TABLE 5. Signature generated by CCM.

Worm	Signature
Apache-Knacker	\xFF\xBF \r\nHost: \r\nHost:
ATPhttpd	\x9E\xF8
TSIG	\xFF\xBF \x00\x00\xFA

TABLE 6. Signature generated by PolyTree.

Worm	Signature
Apache-Knacker	GET HTTP/1.1\r\n : \r\nHost: \r\n : \r\nHost: \xFF\xBF \r\n
ATPhttpd	GET / \xFF\xBF HTTP/1.1\r\n
TSIG	\xFF\xBF \x00\x00\xFA

TABLE 7. DNN model generated signatures and accuracy.

Worm	Signature	*Accuracy*
Apache-Knacker	GET HTTP/1.1\r\n : \r\nHost: \r\n : \r\nHost: \xFF\xBF \r\n	98.2%
ATPhttpd	GET \x9e\xf8 HTTP/1.1\r\n	98.5%
TSIG	\xFF\xBF \x00\x00\xFA	99.3%

where w_i represents the ith byte in the predicted signatures, and $p(w_i|w_1 w_2 \ldots w_{i-1})$ indicates the probability of w_i. As a result, the smaller the perplexity is, the better the performance of the model is.

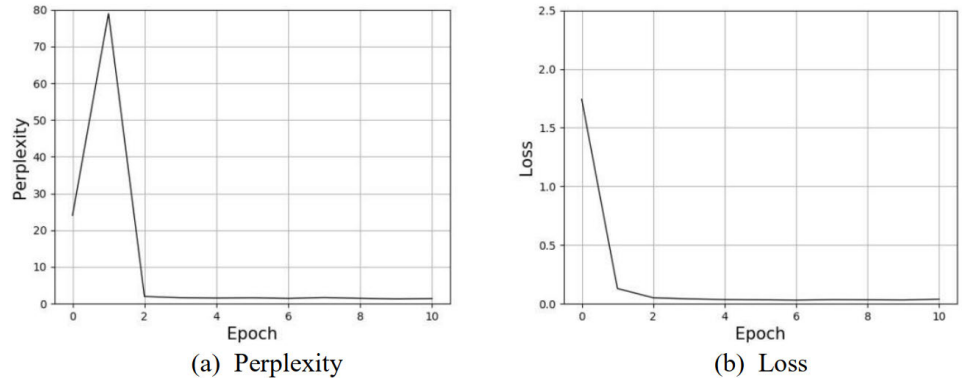
In addition, accuracy is used to evaluate the effect of the DNN algorithm on extracting worm signatures. The accuracy can be obtained by matching the signatures extracted by the DNN algorithm and the original signatures. Accuracy's formula is as follows:

$$Accuracy = \frac{matchTrue}{matchTrue + matchFalse} \tag{25}$$

where *matchFalse* denotes the number of bytes generated by the DNN algorithm that does not match the original signatures, *matchTrue* represents the number of bytes generated by the DNN algorithm that matches the original signatures.

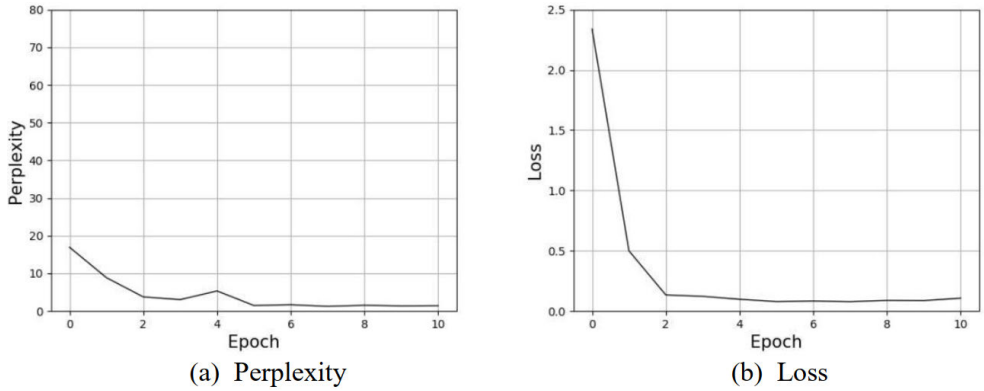
1) SIGNATURE GENERATION OF KNOWN WORMS

We use signatures that Polygraph [11], CCM [4], PolyTree [5] extracted, and their corresponding polymorphic worm payloads as training data to evaluate the performance of DNN model by the accuracy. The results are shown in Table7, Table 8, and Table 9. The accuracy of Apache-Knacker, ATPhttpd, and TSIG is 98.7%, 98%,99.3%, respectively.



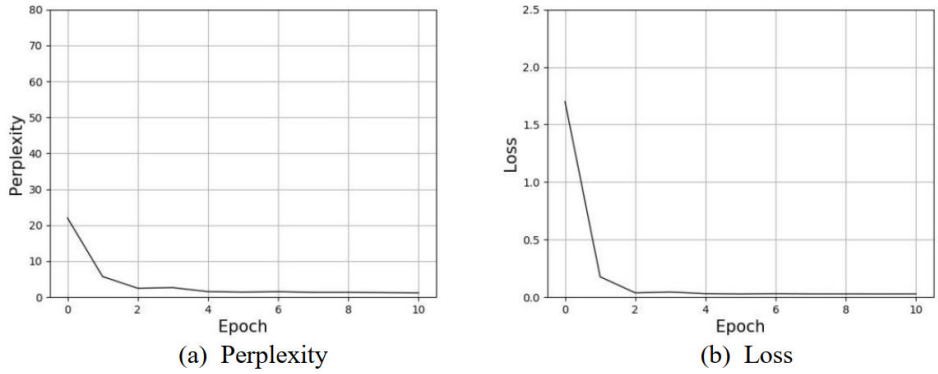
(a) Perplexity (b) Loss

FIGURE 5. Worm signature automatic generation training results based on polygraph.



(a) Perplexity (b) Loss

FIGURE 6. Worm signature automatic generation training results based on CCM.



(a) Perplexity (b) Loss

FIGURE 7. Worm signature automatic generation training results based on PolyTree.

TABLE 8. DNN model generated signatures and accuracy.

Worm	Signature	*Accuracy*
Apache-Knacker	\xFF\xBF \r\nHost: \r\nHost:	99.1%
ATPhttpd	\x9E\xF8	98.1%
TSIG	\xFF\xBF \x00\x00\xFA	99.0%

TABLE 9. DNN model generated signatures and accuracy.

Worm	Signature	*Accuracy*
Apache-Knacker	GET HTTP/1.1\r\n : \r\nHost: \r\n : \r\nHost: \xFF\xBF \r\n	98.7%
ATPhttpd	GET / \xFF\xBF HTTP/1.1\r\n	98.0%
TSIG	\xFF\xBF \x00\x00\xFA	99.3%

It suggests that the DNN model has a good performance for accurate signature generation. The selection of proper signatures has little influence on the signature generation accuracy.

Figure 5, Figure 6, and Figure 7 show the loss and perplexity curve of three signatures as the epoch progresses on the training dataset. Considering the loss and perplexity of training, we observe that both of them converge quickly at the

TABLE 10. Worm detection results.

DNN Model	Signature	*Accuracy*
	Unknown Worm Detection Results	
Polygraph-based DNN model	GET HTTP/1.1\r\n : \r\nHost: \r\n : \r\nHost: \xff\xbf \r\n	97.5%
CCM-based DNN model	\xff\xbf \r\nHost: \x00\x00\xfa	96.5%
PolyTree-based DNN model	GET HTTP/1.1\r\n : \r\nHost: \xff\xbf \r\n	97%
	Worm Detection Results of the hybrid method[32]	
Polygraph-based DNN model	GET HTTP/1.1\r\n : \r\nHost: \r\n : \r\nHost: \xff\xbf	93.7%
CCM-based DNN model	\xff\xbf \x00\x00\xfa	92.1%
PolyTree-based DNN model	GET HTTP/1.1\r\n : \r\nHost: \xff\xbf	93.5%

beginning of training and stay in a small range. That means the DNN model can be trained easily.

2) SIGNATURE GENERATION OF UNKNOWN WORMS

We used 10-fold cross-validation to train the model with three signatures extracted by Polygraph, CCM, PolyTree, and also used CodeRed worm payloads not previously utilized in training to explore the utility of our approach to detect unknown polymorphic worms. The payloads of unseen payloads (CodeRed worm payloads) and dataset payloads had different distribution, and the generated signatures are shown in Table 10. To test the accuracy of signatures, we combined the CodeRed worm packets with the real traces randomly. Then Snort implemented by the generated signatures monitors the network the combined traces are replayed to. The corresponding accuracy is 97.5%,96.5%,97%. That means that the signatures generated by the DNN model have a good performance.

Although we have a private worm payload dataset of a company(3000 worm payloads), we cannot show the generated signatures due to the privacy. We wrote the Snort rules based on the generated signatures of the private payload dataset and conducted the same experiment with CodeRed worm payloads above. The accuracy is 95.1%, which means that the DNN model can construct high-quality signatures of unknown worms.

We experimented with the DNN model which had less than four hidden layers, and the results were very bad. On the contrary, if we selected the model which had more than four layers, the results were almost the same. So we selected the DNN model with four hidden layers. The results are very similar to the results shown in Table 9 and 10, thus we do not show the results here again.

3) COMPARISON

To test the feasibility of our proposed model, we compared the DNN model with the hybrid method [31]. The results are shown in Table 10. The DNN model outperforms the hybrid method.That means that it holds great superiority in generating signatures of worms.

VII. CONCLUSION

In this paper, we propose a worm detection system based on deep learning. It includes a CNN-based worm detection module and a DNN-based worm signature automatic generation module. In the CNN-based worm detection module, we propose three data preprocessing methods: frequency processing, frequency weight processing, and differential processing, and use the CNN model to detect worms based on the three kinds of preprocessed data. In the DNN-based worm signature automatic generation module, worm payloads are input into the trained DNN model, and worm signatures are generated by the proposed Signature Beam Search algorithm. In the experiments, we firstly analyzed the effects of different elements on the worm detection performance. Then we analyzed the effects of different signatures on the automatic generation of worm signatures. Experiments show that the generated signatures have both low false positives and false negatives. Further research is needed, for example, on the signature generation of other attacking methods.

REFERENCES

[1] *2016-2017 Global Cyberspace Security Roundup*, Guoxin Secur. Res. Inst., 2017.

[2] R. Kaur and M. Singh, "A survey on zero-day polymorphic worm detection techniques," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1520–1549, 3rd Quart., 2014.

[3] S. A. Aljawarneh, R. A. Moftah, and A. M. Maatuk, "Investigations of automatic methods for detecting the polymorphic worms signatures," *Future Gener. Comput. Syst.*, vol. 60, pp. 67–77, Jul. 2016.

[4] B. Bayoglu and I. Sogukpınar, "Graph based signature classes for detecting polymorphic worms via content analysis," *Comput. Netw.*, vol. 56, no. 2, pp. 832–844, Feb. 2012.

[5] Y. Tang, B. Xiao, and X. Lu, "Signature tree generation for polymorphic worms," *IEEE Trans. Comput.*, vol. 60, no. 4, pp. 565–579, Apr. 2011.

[6] A. Mondal, S. Paul, A. Mitra, and B. Gope, "Automated signature generation for polymorphic worms using substrings extraction and principal component analysis," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res. (ICCIC)*, Dec. 2015, pp. 1–4.

[7] R. Eskandari, M. Shajari, and A. Asadi, "Automatic signature generation for polymorphic worms by combination of token extraction and sequence alignment approaches," in *Proc. 7th Conf. Inf. Knowl. Technol. (IKT)*, May 2015, pp. 1–6.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*. Red Hook, NY, USA: Curran Associates, vol. 2012, pp. 1097–1105.

[9] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," 2014, *arXiv:1404.2188*. [Online]. Available: http://arxiv.org/abs/1404.2188

[10] D. Zhu, H. Jin, Y. Yang, D. Wu, and W. Chen, "DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017, pp. 438–443.

[11] K. Ozkan, S. Isik, and Y. Kartal, "Evaluation of convolutional neural network features for malware detection," in *Proc. 6th Int. Symp. Digit. Forensic Secur. (ISDFS)*, Mar. 2018, pp. 1–5.
[12] C. H. Kim, E. K. Kabanga, and S.-J. Kang, "Classifying malware using convolutional gated neural network," in *Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2018, pp. 40–44.
[13] S. M. A. Sulieman and Y. A. Fadlalla, "Detecting zero-day polymorphic worm: A review," in *Proc. 21st Saudi Comput. Soc. Nat. Comput. Conf. (NCC)*, Apr. 2018, pp. 1–7.
[14] D. Nahmias, A. Cohen, N. Nissim, and Y. Elovici, "Deep feature transfer learning for trusted and automated malware signature generation in private cloud environments," *Neural Netw.*, vol. 124, pp. 243–257, Apr. 2020.
[15] P. Szynkiewicz and A. Kozakiewicz, "Design and evaluation of a system for network threat signatures generation," *J. Comput. Sci.*, vol. 22, pp. 187–197, Sep. 2017.
[16] F. Wang, S. Yang, D. Zhao, and C. Wang, "An automatic signature-based approach for polymorphic worms in big data environment," in *Proc. Int. Conf. Netw. Netw. Appl.*, Oct. 2019, pp. 223–228.
[17] Y. Afek, A. Bremler-Barr, and S. L. Feibish, "Zero-day signature extraction for high-volume attacks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 691–706, Apr. 2019.
[18] S. N. Nguyen, V. Q. Nguyen, J. Choi, and K. Kim, "Design and implementation of intrusion detection system using convolutional neural network for DoS detection," in *Proc. 2nd Int. Conf. Mach. Learn. Soft Comput.*, 2018, pp. 34–38.
[19] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1222–1228.
[20] S. Venkatraman, M. Alazab, and R. Vinayakumar, "A hybrid deep learning image-based analysis for effective malware detection," *J. Inf. Secur. Appl.*, vol. 47, pp. 377–389, Aug. 2019.
[21] W. Zhong and F. Gu, "A multi-level deep learning system for malware detection," *Expert Syst. Appl.*, vol. 133, pp. 151–162, Nov. 2019.
[22] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *Innovations in Machine Learning*. Berlin, Germany: Springer, 2006, pp. 137–186.
[23] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *Comput. Sci.*, 2015.
[24] [Online]. Available: http://torch.ch/
[25] *C. CAN-2003-0245. Apache Apr-Psprintf Memory Corruption Vulnerability*. [Online]. Available: http://www.securityfocus.com/bid/7723/discussion/
[26] *SANS Institute: Lion Worm*. [Online]. Available: http://www.sans.org/y2k/lion.htm
[27] (Oct. 2011). *ATPhttpd Remotely Exploitable Buffer Overflow*. [Online]. Available: http://secunia.com/advisories/7293/
[28] [Online]. Available: http://openmalware.org/search.cgi?search=codered
[29] M. Amin, T. A. Tanveer, M. Tehseen, M. Khan, F. A. Khan, and S. Anwar, "Static malware detection and attribution in Android byte-code through an end-to-end deep system," *Future Gener. Comput. Syst.*, vol. 102, pp. 112–126, Jan. 2020.
[30] S. Alam, S. A. Alharbi, and S. Yildirim, "Mining nested flow of dominant APIs for detecting Android malware," *Comput. Netw.*, vol. 167, Feb. 2020, Art. no. 107026.
[31] S. Kaur and M. Singh, "Hybrid intrusion detection and signature generation using deep recurrent neural networks," *Neural Comput. Appl.*, vol. 32, no. 12, pp. 7859–7877, Jun. 2020.

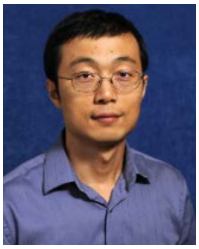
HANXUN ZHOU was born in 1981. He received the master's and Ph.D. degrees in computer application technology from Northeastern University, in 2006 and 2009, respectively. He is currently an Associate Professor with Liaoning University. His research interest includes network security, especially malcode.

YESHUAI HU was born in 1994. He received the B.D. degree in science from Shanxi Normal University, in 2018. He is currently pursuing the master's degree with Liaoning University. His research interests include digital economy and cybersecurity.

XINLIN YANG was born in 1996. He graduated from the Shandong University of Technology, in 2019. He is currently pursuing the master's degree with Liaoning University. He followed Dr. Zhou Hanxun to learn the direction of network security based on machine learning and so on.

HONG PAN was born in 1979. He received the Ph.D. degree in economics from Liaoning University, in 2016. He is currently an Associate Professor with Liaoning University. His research interests include digital economy, big data management, distributed data management, and uncertain data management.

WEI GUO was born in China, in 1983. She received the B.S., M.S., and Ph.D. degrees from Northeast University, Shenyang, Liaoning, China, in 2006, 2008, and 2011, respectively. Since her graduation, she has been working with Shenyang Aerospace University, China, where she is currently an Associate Professor. Her research interests include artificial intelligence, machine learning, and image processing.



CLIFF C. ZOU (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the Department of Automation, University of Science and Technology of China, in 1996, 1999, and 2005, respectively. He is currently an Associate Professor with the University of Central Florida. His research interests include computer and network security, computer networking and network modeling, and performance evaluation.