Southeast Asian J. of Sciences Vol 6, No 2, (2018) pp. 147-159

COMBINATION OF MULTI-CHANNEL CNN AND BiLSTM FOR HOST-BASED INTRUSION DETECTION

Nguyen Ngoc Diep, Nguyen Thi Thanh Thuy
and
Pham Hoang Duy

Department of Information Technology
Posts and Telecommunications Institute of Technology (PTIT)
Hanoi, Vietnam
e-mail: diepnguyenngoc@ptit.edu.vn

Abstract

A significant increase of intrusion events over the years imposes a challenge on the robust intrusion detection system. In a computer system, execution traces of its programs can be audited as sequences of system calls and provide a rich and expressive source of data to identify anomalous activities. This paper presents a deep learning model, which combines multi-channel CNN and bidirectional LSTM (BiLSTM) models, to detect abnormal executions in host-based intrusion detection systems. Multi-channel CNN with word embedding can in large extent be used to extract relationship features of system calls. Meanwhile, BiLSTM enables our model to understand the context of system call sequences thanks to capturing long-distance dependencies across the sequences. The integration of these two models leads to the efficient and effective detection of abnormal behaviors of a system. Experiment results on ADFA-LD dataset show that our approach outperforms other methods.

Key words: Intrusion detection, ADFA-LD dataset, multi-channel CNN, BiLSTM.
2010 AMS Mathematics classification: 91D10, 68U115, 68U35, 68M14, 68M115, 68T99.

1 Introduction

In recent years, attacks on organization computer systems are increasing in volume, severity, and duration. In such scenarios, intrusion detection plays a critical role in preventing malicious activities on the systems. An intrusion detection system (IDS) can effectively discriminate between malicious and benign system activities. Different types of IDS are defined based on the scope of intrusion monitoring and methodological approaches.

According to the scope of intrusion monitoring, there are two types of IDS namely host-based and network-based systems. The network-based intrusion detection systems monitor network traffic between hosts to protect the system from network threats. This type of IDS collects information from network packets and analyses the content in order to detect malicious activities from the network. Meanwhile, host-based detection systems monitor the activities on a single system. It analyses collected information from the host system like event logs or system logs in order to identify vulnerability exploits against a target software or a computer system.

Depending on methodological approaches, IDS can be classified into signature-based and anomaly-based. Signature-based approaches operate by matching captured behaviors against known intrusion patterns. This approach is very accurate (no false alarm) but cannot detect unknown patterns. On the other hand, the anomaly-based approach can learn from existing knowledge to differentiate abnormal and normal behaviors. Thus, it can potentially detect unseen and novel intrusions. However, it may result in higher false alarm rates.

In this work, we focus on anomaly detection in host-based IDS. One important approach in detecting host-based intrusion exploits the sequences of system calls recorded during the executions of processes in the system. These sequences can be fed into a learning model as features so that malicious behaviors of processes can be recognized. However, these sequences differ from each other in their lengths (depending on running processes), execution contexts and orders when calling system API. These differences lead to difficulties in developing learning models that predict and classify malicious processes. This is similar to the case of a sequence of words in a text sentence, which may have word-level features, phrase-level features, and context features of the sequence/sentence. It is difficult to identify all of them simultaneously.

Previous works of anomaly-based host intrusion detection methods have been focused on the feature engineering approach. This approach tries to identify meaningful features manually, therefore, it is hard to capture many useful features of system call sequences. In recent years, deep learning has achieved remarkable success in many applications like natural language processing (NLP), image processing, etc. Due to the similar formation between a sequence of system calls in a trace and a sequence of words in a sentence, recent works like [10] used deep neural language models to detect anomaly sequences of sys-

tem calls. They consider system call sequences as instances of the language for communication between users (or programs) and the system. In this view, system calls and their sequences correspond to words and sentences in natural languages. This deep neural language model based approach has shown significant improvement in detecting abnormal activities.

Motivated by the deep neural language model based approach, this work formalizes traces of system calls during process executions as natural sentences and constructs deep learning models to improve the performance of intrusion detection. In particular, the proposed approach utilizes the technique of word embedding to represent the sequence of system calls and feed this output to a deep learning model combining multi-channel CNN and bidirectional LSTM (BiLSTM) to detect malicious activities of running processes. Multi-channel CNN with word embedding can be used to extract system call relation features. BiLSTM help to understand full context of system call sequences by capturing long-distance dependency across the sequence in both left and right directions. The integration of these two advanced models contributes to efficient and effective detection of abnormal behaviors in the system. The performance of the proposed model is evaluated against ADFA-LD dataset [4]. Furthermore, the paper investigates several deep learning models in the field to better illustrate the performance of the proposed model.

2 Related Work

Previous works of detecting anomaly system call sequences have been focused on the feature engineering approach with two major categories including short sequence-based and frequency-based [6]. The authors of [7] propose a simple method relied on sliding windows to extracts a fixed-size sequence of system calls. Then, this sequence is used as a feature vector. This method is just efficient to monitor short sequences of system calls but may not be capable to handle against sufficiently long sequences. In [5], the authors construct a semantic model (dictionary) for the short sequences in the forms of "word" and "phrase". Based on this dictionary, they evaluate hidden Markov model, extreme learning machine, and one-class SVM algorithms. The results is rather good with false positive rate of 15%, extreme learning machine obtains the detection accuracy of 90% and one-class SVM algorithm of 80%. However, this method is very time-consuming.

The paper [1] represents system call traces by sliding several n-grams of variable length over the system call traces and computing the occurrences of these n-grams in the traces. Therefore, the orders of system calls are maintained while providing novel features for classification models. ADFA dataset has confirmed that 6-grams for representing the system traces grants the best performance at a low level of false positive rate against other representations in-

cluding term frequency, inverse document frequency, and sequences. Amongst classifier models, one-class SVM model claims the best with the accuracy of 95% and the true positive rate of 87% at the low false positive rate of 5%.

Another work by Miao Xie et al. [17] applies the concept of frequency of system call traces with one-class SVM but still suffers the bad performance with the average accuracy of around 70%. In order to represent traces of system calls as vectors, this paper combines Boolean model, where every system call is counted for its existence in the traces of system calls, and Modified Vector Space model, which can preserve the relative orders of system calls in the traces by considering a sequence of system calls as a unique term instead of a single system call. Furthermore, the modified vector model tackles with the occurrences of a new sequence of system call during the testing phase by including a system call whose number higher than any known one in the training phase. Given the vector representation of traces, the paper has investigated the performances of typical classifiers including Naïve Bayes, support vector machine, k-mean, and decision tree. ADFA-LD dataset shows that increasing term sizes improves the performance of SVM and Naïve Bayes models in both accuracy and false positive rate. The average accuracy of the investigated models is around 80%. Naïve Bayes model achieves the best with the false positive rate at around 10%.

The authors of [12] improve a detection model, which employs a combined first order Markov-Bayes algorithm, by applying domain knowledge, boost technique, and more complex and higher-order representation of Markov chains. The proposed model has investigated by ADFA-WD dataset. Adding specific knowledge of the system call traces as the whole system approach produces the best performance of both models using boost technique and higher-order Markov chain with accuracy ranging from 88% to 97% while F1 factor above 92%. But, these models suffer the high false alarm rate accounting for around 29%.

Recently, deep neural language model based approach has gained remarkable results in detecting abnormal system call sequences as it can capture call-level features, phrase-level features and context features in call sequence. The paper [10] proposes an LSTM based language model-based method for detecting abnormal programs running in computer systems. The model considers system calls and their traces as basic elements of the language roughly corresponding to words and bags of words created by the programs. The dependencies amongst these elements are learned by LSTM so that the proposed model can identify a malicious pattern of computer programs. The detection rate of the model reaches 90%, which is comparable to existing models such as the hidden Markov model, but rather low false alarm rate (at 16%). In [13], the author uses RNN and GRU for intrusion detection and report a good detection rate at 98.3%. In [2], Chawla et al. apply a similar concept using stacked CNN over GRU and achieve better performance with AUC of 0.81.

Our work also follows deep neural language model based approach to detect abnormal traces. In particular, we combine multi-channel CNN and bidirectional LSTM models because these two models are capable of capture more useful features from sequences of system calls compared to the original models including CNN and LSTM/GRU.

3 Background

3.1 Convolution Neural Network

Supposed that a sequence of system call of length is s, a matrix d x s has every row that is a d-dimension word embedding vector of each system call (using an embedding layer). Given a matrix S of these sequences, CNN performs convolution on this matrix via linear filters, which is a weight matrix W of length d and region size h. For an input matrix $S \in \mathbb{R}^{d \times s}$, a feature map vector $O = [o_0, o_1, ..., o_{s-h}] \in \mathbb{R}^{s-h+1}$ of the convolution operator with a filter W is obtained by applying repeatedly W to sub-matrices of S:

$$o_i = W \cdot S_{i:i+h-1} \tag{1}$$

where i = 0, 1, 2, . . . ,s-h; $S_{i:j}$ is the sub-matrix of S from row i to j.

Each feature map is then fed to a pooling layer to generate potential features. This layer effectively combines several values into a single one and helps to decrease the chance of over-fitting because those very particular are discarded thanks to the pooling process. For example, max pooling uses the maximum value from the feature maps to capture the most important feature v. Besides that, max pooling is the most commonly used strategy.

$$v = \max_{0 \leqslant i \leqslant s-h} \{o_i\} \tag{2}$$

In this work, we apply multiple filters with variant region sizes in order to obtain multiple max pooling values. After pooling, these pooling values from feature maps are concatenated into a CNN feature. To make a connection to these values, a dense layer is deployed to synthesize a high-level feature from the CNN features.

3.2 Long Short Term Memory model

The architecture of a recurrent neural network (RNN) is suitable for processing sequential data. However, a simple RNN is usually difficult to train because of the gradient vanishing problem [8]. In [9], the authors propose LSTM architecture which utilizes a memory cell preserving its state over a long period of time and non-linear gating units regulating information flow into and out

of the cell. This cell let LSTM have ability to capture efficiently long-distance dependencies of sequential data without suffering the exploding or vanishing gradient problem. An advantage of LSTM over RNNs and other sequence learning methods is relative insensitivity to gap length.

Sequences of system calls of variable length are transformed to fix-length vectors by recursively applying a LSTM unit to each input call x_t of a sequence and the previous step h_{t-1}. At each time step t, LSTM unit with l-memory dimension defines 6 vectors in $\mathbb{R}^l$: input gate i_t, forget gate f_t, output gate o_t, *tanh* layer for input transformation u_t, memory cell c_t and hidden state h_t as follows:

Gates:

$$i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \tag{3}$$

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{4}$$

$$o_t = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \tag{5}$$

Input transformation:

$$u_t = tanh\left(W_u x_t + U_u h_{t-1} + b_u\right) \tag{6}$$

State update:

$$c_t = f_t \otimes c_{t-1} + i_t \otimes u_t \tag{7}$$

$$h_t = o_t \otimes tanh(c_t) \tag{8}$$

where x_t is the input vector; *W, U, b* are layer parameters; σ is a sigmoid function.

Intuitively, the forget gate makes a decision of which previous information in the memory cell should be forgotten, while the input gate controls what new information should be stored in the memory cell. And, the output gate decides the amount of information from the internal memory cell should be exposed. These gate units enable LSTM model to retain significant information over multiple time steps.

In this paper, we adopt the bidirectional LSTM, which extends the traditional LSTMs in order to improve the performance on sequence classification problems. For those problems where all time-steps of the input sequence are available, the bidirectional LSTM trains two instead of one LSTM on the input sequence. The first on the input sequence is trained as-is and the second on a reversed copy of the same input. That can provide additional context to the network and results in faster and even comprehensively learning on the problem.

4 Proposed Method

We assume that the host system generates a finite number of system calls during its programs' executions. Considering traces of system calls created by executing computer programs as sentences of a document (where each system call is a word), the problem of detecting intrusion can be viewed as the problem of classifying a sentence into normal or abnormal. This is similar to the problem of sentiment classification into two classes: positive and negative, a well-known problem in natural language processing (NLP). Modern approaches to NLP often come with word embeddings, a vector representation technique, to obtain the best performance. Thus, this work applies this technique to represent sequences of system calls by integrating a word embedding layer into our proposed model. The vectors of these sequences are presented as input for deep learning processing in the next stages of our model.

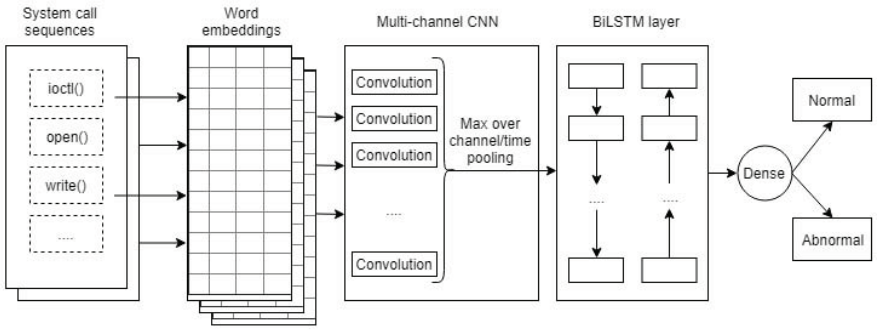

Figure 1: The proposed architecture

The deep learning processing combines a multi-channel CNN and BiLSTM models. Given an input, CNN model can capture local dependencies between neighbor words or system calls. However, because of the limitation of filter length in CNN model, it is hard to learn all dependencies of the whole sequence. This can be tackled by using multi-channel CNN model [11] which allow concatenating local relationship values.

LSTM model enables a cell to maintain information over a long period of time. Therefore, a feature vector constructed by LSTM can carry overall dependencies of the input vector. LSTM can capture the left context of the input vector but bidirectional LSTM (BiLSTM) can even do more than that. BiLSTM can process both of the left and right contexts to see how well an element can fit in the input vector. The advantages of both multi-channel CNN and BiLSTM can be combined to enhance the classification performance.

Figure 1 shows the architecture of the proposed model. Given a trace of system calls, our proposed model generates word-embedding vectors via an

embedding layer. These embedding vectors are then fed to multi-channel CNN model to create CNN feature vectors. Next, these CNN vectors are converted into LSTM feature vectors thanks to BiLSTM layer. Finally, these LSTM vectors are classified by a neural network. The proposed model can improve the classification performance by capturing both local and global dependencies of a sequence of system calls thanks to multi-channel CNN and BiLSTM models. In training phase, the proposed model adopts *rmsprop* algorithm [14] to learn model parameters.

In details, our multi-channel CNN derives that of [11] and consists of 4 convolution stacks with different filter lengths. Each stack consists of three layers including one convolution, one dropout, and one max-pooling. Essentially, one feature is extracted from one filter. Multiple filters with varying window size are deployed in our model in order to acquire multiple features. These feature vectors are merged together to form a single vector and passed to two convolution layers with max-pooling. After these vectors are processed by BiLSTM layer, the output vectors are supplied to a fully connected neural network layer with *softmax* function so that these vectors are labeled with a probability distribution. The dropout technique can support regularisation and reduce the over-fitting by avoiding training nodes on all training data. As a result, the technique enables the network to learn more robust features. The convolutional layer uses rectified linear unit (ReLU) as an activation function which applies a non-linear operation to a feature map as replacing all negative values by zero. ReLU function is preferable over other activation functions thanks to its better performance in most situations.

Regarding multi-channel CNN, filter lengths of 4, 6, 8, and 10 are deployed. The number of filters for all convolution layers are 128. The size of memory cells used in BiLSTM stage accounts for 500. BiLSTM applies the dropout technique with the recurrent dropout of 0.2. The dimension of the last layer of the fully connected neural network is set to 128. In addition, the probability of selecting units is at 0.5 for the dropout layer. The network in the proposed model is trained using mini-batches having the size of each mini-batch of 50. The negative log likelihood is minimized by *rmsprop* optimizer provided in Keras with learning ratio of 0.01.

5 Experiment Evaluation

5.1 Dataset

ADFA-LD dataset [4] is used to examine the performance of our proposed model. This is the most current and expressive dataset created by the University of New South Wales and contains execution traces in Linux environment (Ubuntu 11.04) equipped with typical applications such as FTP, SSH, MySQL, Apache, PHP and web-based tools. Ubuntu operating system is fully updated

and the applications are started with default ports. This configuration represents a common working environment of a local server with some known exposures. That is the system is likely to suffer several real types of exploitations. The attacks being replicated in ADFA dataset, which are likely required the intermediate hacking level, includes FTP Hydra, SSH Hydra, client-side poisoned executable, Tiki Wiki vulnerability exploit, and PHP remote file inclusion vulnerability.

ADFA-LD aims for anomaly detection systems and constitutes three data groups that are raw traces of system calls recorded during operations of the environment. The traces' sizes in training data are within 300 bytes and 6KB while those of testing data are between 300 bytes and 10KB. There are about 800 and 4300 traces in training and testing data respectively. Meanwhile, the attack data contains about 750 traces of abnormal executions. The overall average length of all traces per system call in the dataset is about 460, which is quite below that of testing data (485) and well above that of training and attack data (369 and 425 correspondingly) as shown in Fig. 2.

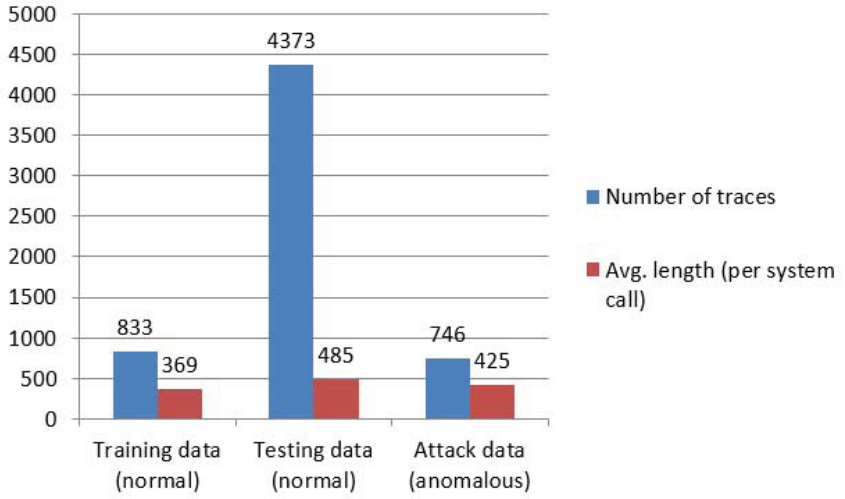

Figure 2: Number of traces and average length per system call for ADFA-LD dataset

5.2 Experimental Settings

5.2.1 Evaluation metrics

Overall accuracy is the simple measurement, which is often used in evaluating classification performance. This measurement accounts for the proportion

between the correct classified elements and the total elements being classified. However, due to the data imbalance among the classes required to be classified, this measurement is not really efficient and appropriate [15, 16]. In this situation, the measurement metrics are composed of precision, recall and F1 (harmonic mean). The precision presents the rate of correct classification/prediction of a class, while the recall illustrates the rate of actual correct classification/prediction. F1 presents the average of both precision and recall, therefore, it facilitates the performance comparison among classification models.

These measurements are defined as following formulae:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{12}$$

where TP (true positives): correct classifications of positive cases; FP (false positives): incorrect classifications of positive cases into negative class; FN (false negatives): incorrect classifications of negative cases into positive class.

In order to evaluate the performance of classification models, 10 folds cross-validation is used in the experiment. This validation technique divides the dataset into 10 parts such that 9 parts is employed for training and the last one is for testing. This technique is repeatedly applied until all of 10 parts are passed through the model. The measurements are computed by applying average functions.

5.2.2 Parameters of the models

The deep neural network in our proposed model is built and trained by Keras library [3] and its parameters are described in Section 3. Regards to LSTM based model, CuDNNLSTM module is preferred in order to improve the execution speed of our proposed model. Because of the imbalance in the dataset, our model applies weighted parameters to interested classes. This technique assigns a weight value to the classes substantially having more instances than the others in order to improve the classification performance. This technique is facilitated by *class_weight* parameter in Keras library [3].

5.3 Evaluation

We compare our proposed model against the following models, namely CNN without word embedding, multi-channel CNN, and BiLSTM. As shown in Table 1, the performance of our model claims the best with the scores of 97% for every measurement and followed by a multi-channel CNN model and BiLSTM model. This result also confirms that the combination of these two models can improve the overall classification performance. Interestingly, the CNN model is not far behind multi-channel CNN with the difference of about 1% in every measurement. Furthermore, the recall of CNN is equally good as that of multi-channel CNN.

Table 1: Experimental results on ADFA-LD dataset

Method	**Accuracy**	**Precision**	**Recall**	**F1**
One-class SVM [17]	0.7	-	-	-
RNN [13]	0.98	1.0	0.71	0.83
CNN without word embedding	0.88	0.86	0.87	0.87
CNN	0.94	0.94	0.94	0.94
Multi-channel CNN	0.95	0.96	0.94	0.95
BiLSTM	0.96	0.96	0.96	0.96
Our proposed	0.97	0.97	0.97	**0.97**

With respect to merely accuracy and precision, RNN model in [13] performs the best with 98% and 100% respectively but the recall and F1 are rather low (at 71% and 83% correspondingly) compared to the others. That means this model suffers a high rate of incorrect classification of abnormal traces.

Existing models using traditional models [5, 1, 17] can achieve the top accuracy of 95% with the exception of [12], which provides 97%. Among these models, only [12] claims F1 reaches about 92% but it requires adding domain specific knowledge into the model. As a result, deep learning models provide better overall performance but do not require human intervention. Table 1 shows that the top 4 models achieve significantly high but balanced results. That means these models provide a low rate of false alarm (high precision) and high rate of correctly detecting intrusion (high recall).

6 Conclusion

In this paper, we introduce a novel model which integrates advantages of multi-channel CNN and BiLSTM for detecting intrusion in host-based environments. With the support of word embeddings and the combination of the two advanced models, our proposed method can capture both rich local information within sequences of system calls and long-distance dependency accross sequences in

the classification process, therefore, achieves the best performance in detecting intrusion. Our investigation on ADFA-LD dataset confirms the superiority of deep learning based models over the classical classification methods in detecting abnormal execution traces.

In the future, we will attempt to apply some enhanced techniques in NLP to improve our model like attention mechanism, some other word embedding techniques and ensemble methods.

References

[1] Bhavesh Borisanya and Dhiren Pate,. *Evaluation of Modifed Vector Space Representation Using ADFA-LD and ADFA-WD Datasets,* Journal of Information Security, **6**(2015), 250-264.

[2] Ashima Chawla, Brian Lee, Sheila Fallon, and Paul Jacob, *Host based Intrusion Detection System with Combined CNN/RNN Model,* Proceedings of Second International Workshop on AI in Security, (September), 2018.

[3] François Chollet, *Keras: Deep Learning library for Theano and TensorFlow.* GitHub Repository, (2015) 1–21.

[4] Gideon Creech and Jiankun Hu, *Generation of a new IDS test dataset: Time to retire the KDD collection,* In IEEE Wireless Communications and Networking Conference, WCNC, 2013.

[5] Gideon Creech and Jiankun Hu, *A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns,* IEEE Transactions on Computers, **63**(4) (2014), 807–819.

[6] Stephanie Forrest, Steven Hofmeyr, and Anil Somayaji, *The evolution of system-call monitoring,* In Proceedings - Annual Computer Security Applications Conference, ACSAC, 2008.

[7] Stephanie Forrest, Steven A Hofmeyr, Anil Somayaji, and Thomas A Longstaff, *A sense of self for unix processes,* In Security and Privacy, Proceedings of 1996 IEEE Symposium, IEEE (1996), 120-128.

[8] C. Goller and A. Kuchler. *Learning task-dependent distributed representations by backpropagation through structure,* In Proceedings of International Conference on Neural Networks (ICNN'96), **1** (1996), 347–352.

[9] Sepp Hochreiter and Jürgen Schmidhuber. *Long Short-Term Memory.* Neural Computation, 1997.

[10] Jaehyun Jihyun Kim, Jaehyun Jihyun Kim, Huong Le Thi Thu, and Howon Kim, *Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection*, 2016 International Conference on Platform Technology and Service (PlatCon), pages 1–5, 2016.

[11] Yoon Kim, *Convolutional neural networks for sentence classification.* arXiv preprint arXiv:1408.5882, 2014.

[12] Tarik Mouttaqi, Tajjeeddine Rachidi, and Nasser Assem., *Re-Evaluation of Combined Markov-Bayes Models for Host Intrusion Detection on the ADFA Dataset,* (September 2017), 1044–1052.

[13] Razvan Pascanu, Jack W. Stokes, Hermineh Sanossian, Mady Marinescu, and Anil Thomas, *Malware classification with recurrent networks,* In ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2015.

[14] Sebastian Ruder, *An overview of gradient descent optimization algorithms,* arXiv preprint arXiv:1609.04747, 2016.

[15] Yuchun Tang, Yan Qing Zhang, and Nitesh V. Chawla, *SVMs modeling for highly imbalanced classification,* IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, **39**(1) (2009), 281–288.

[16] Konstantinos Veropoulos, Colin Campbell, Nello Cristianini, and Others, *Controlling the sensitivity of support vector machines,* Proceedings of the international joint conference on artificial intelligence, (1999), 55–60.

[17] Miao Xie, Jiankun Hu, Xinghuo Yu, and Elizabeth Chang, *Evaluating Host-Based Anomaly Detection Systems: Application of the Frequency-Based Algorithms to ADFA-LD,* Network and System Security, pages 542–549, 2014.