Grasp the Key: Towards Fast and Accurate Host-based Intrusion Detection in Data Centers

Mengtian Gu[1,2], Biyu Zhou[1] (✉), Fengyang Du[1,2], Xuehai Tang[1,2], Wang Wang[1,2], Liangjun Zang[1], Jizhong Han[1], and Songlin Hu[1]

[1] Institute of Information Engineering, Chinese Academy of Sciences, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, China
zhoubiyu@iie.ac.cn

Abstract. With the rapid development of data center facilities and technology, in addition to detection accuracy, detection speed has also become a concern for host-based intrusion detection. In this paper, we propose a DNN model to detect intrusion for host with high accuracy. Along with that, a data reduction method based on SHapley Additive exPlanations (SHAP) is incorperated to reduce the execution time of the DNN model. Extensive evaluation on two well-known public datasets in this field shows that our proposed method can achieve high-efficiency intrusion detection while ensuring high-precision.

Keywords: Intrusion detection · Explainable artificial intelligence · Neural networks.

1 Introduction

In recent years, security threats against computers have become more and more serious. The Host-based Intrusion Detection System (HIDS), as a kind of active protection technology, has been gaining extensive attention in the field of computer security. However, despite considerable efforts in this field, HIDS has encountered the well-known big data challenge brought by the rapid development of data center facilities and technologies [13]. First of all, large data centers require high precision for intrusion detection. Once an exception occurs within the data center, it will spread quickly and finally affect the whole cluster. Furthermore, handling a large number of system call traces has become a basic requirement for modern data centers, which is a big challenge for detection efficiency due to the real-time requirement.

Deep learning has shown its ability to discover potential patterns of big data and achieve high-precision intrusion detection in recent years. However, the high complexity of the network also results in long execution time of the model. For instance, it may take several days to train a deep learning model in a large data center. This is not a problem if the system only needs to be trained once. But in practice, because of new patches and modifications to the system, it is often necessary to train several times to accommodate the new changes. More importantly, the model needs to handle a large number of fine-grained system call traces at the same time when doing intrusion detection. Understandably, this causes a speed limit.

In this paper, we seek to develop a novel intrusion detection method and introduce specialized designs to solve the above challenges. In terms of intrusion detection model, we choose two neural networks: multi-filter CNN and attention-based BiLSTM, which can extract local and global feature representation well. We combine these two neural networks to achieve a high precision. Then we use an XAI method — SHapley Additive exPlanations (SHAP) to obtain the *important decision interval*, so that we can implement model acceleration by data reduction.

We experimented with two publicly available datasets: ADFA-LD and UNM. The results show that our proposed method can greatly reduce execution time of the intrusion detection model and has little impact on precision. To the best of our knowledge, this is the first intrusion detection technology to be customized for big data by using XAI. By accelerating the model execution process, our efforts can make a positive contribution to building a reliable deep learning intrusion detection system in the current big data and cloud computing environment.

The rest of this article is composed as follows. In Section 2, we present a brief background and the motivation; In Section 3, we give a literature review. In Section 4, we describe the proposed method; Two different system call datasets are evaluated in Section 5; Section 6 discusses our conclusion and summarizes the future work.

2 Background and Motivations

With the frequent occurrence of various network security issues, intrusion detection can actively defend against various attacks and has gradually become a research hotspot in the field of computer security. Intrusion Detection Systems (IDS) are devices which monitor systems to detect potential intrusions. They can be divided into Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS). NIDS collects information from the packets of data, then monitors and analyzes network traffic to protect the system from network-based threats. Compared with NIDS, HIDS mainly collects information such as host system calls or logs. It focuses on using these indicators to determine whether the host system has been compromised.

System calls provide the basic interface between the process and the operating system. The system call is referenced when the running process requests the kernel service from the operating system. Thus, it constitutes a trace which describes the behavior of the monitored process. Compared with other data sources for HIDS, system call traces can better reflect differences between normal and abnormal behavior. At present, most HIDS use system call traces as their main source of information. This article also mainly discuss intrusion detection technology based on system calls.

The traditional HIDS mainly performs intrusion analysis on independent hosts installed with independent detection software. However, with the rapid development of data centers and other technologies, the application scenario of HIDS has changed. Processing a large number of system call traces of multiple hosts has become a basic requirement for modern data centers. Therefore, this presents some new challenges to HIDS.

Challenge 1: Detection accuracy — System call traces generated by a large data center are a kind of big data, which are huge and complex. On the other hand, once an internal server is abnormal, this exception will spread quickly and then affect the entire cluster. This may cause considerable damage to the data center. Therefore, we need an intrusion detection technology which can accurately identify each intrusion behavior. In other words, it requires high accuracy and low false alarm rate.

Challenge 2: Execution time — The execution time of HIDS is usually measured by the training time and detection time of the model. For the training time, it is very cumbersome to maintain or update large amounts of traditional HIDS software installed on each host or virtual host in the network. As for the detection time, Fig. 1 shows the detection time of a RNN model with the increase of data volume under a single GPU. With the increase of data volume, the detection time of HIDS will no longer meet the requirement of real-time. One solution is to use multiple GPUs deployed on physical hosts to speed up the detection process. However, it can be expensive and space-consuming.

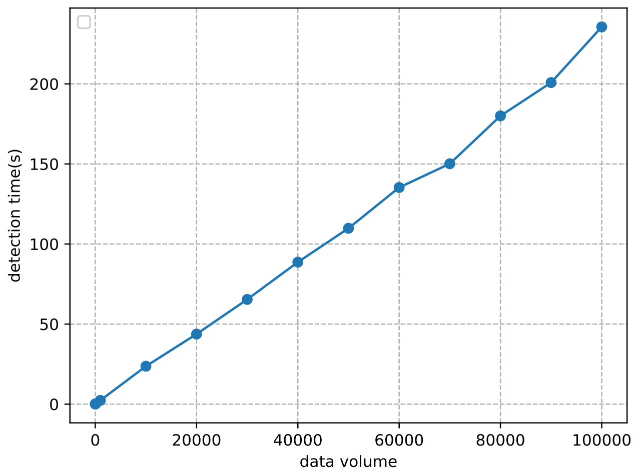


Fig. 1. Detection time of a RNN model with the increase of data volume under a single GPU

3 Related Work

This section briefly discusses the work closely related to this paper.

3.1 Host-based Intrusion Detection

Numerous work and surveys have been published in the area of intrusion detection. In the field of traditional methods, Forrest et al. [6] first introduced a system-call-based method into intrusion detection research works, which is known as "sequence time-delay embedding (STIDE)". This method constructs normal databases to detect anomalies which would take a lot of time to update and maintain. Compared with STIDE, Wang et al. [21] implemented a method based on Bloom filter which takes up less memory. However, the limitation of Bloom Filter also exits as it allows false positives. Pierre-Francois Marteau [16] proposed a new sequence similarity measure to distinguish normal and abnormal sequences. However, it is difficult to obtain dependencies between sequences. Some well-known machine-learning models such as support vector machine (SVM) [1, 9], the k-nearest neighbor algorithm (KNN) [12, 5] and decision trees have been implemented on HIDS. Yet these methods are difficult to adapt to the current distributed computing environment.

Some recent works have used deep learning to improve the performance of HIDS, which has achieved remarkable success in many other fields. Staudemeyer and Trivedi [19] applied LSTM neural network to intrusion detection and achieved preferable results. Ghosh et al. [7] introduced the Elman neural network into HIDS. Wenqi Xie et al. [22] proposed a sensitivity-based LSTM model to design a System-call Behavioral Language (SBL) system for intrusion detection. In conclusion, deep learning may be applicable to HIDS in the big data environment as it has shown its ability to discover potential patterns within big data. But due to the increasing number of system calls being generated, the execution time of deep neural networks can be complex and time consuming. One solution is to use multiple GPUs to speed up the execution process. However, the result presented in the previous section turns out that this solution can be expensive and space-consuming [13].

3.2 Explainable Artificial Intelligence

Explainable artificial intelligence (XAI) is a field of artificial intelligence (AI) which can help us understand how deep learning models learn and why they make such decisions for each input. XAI systems can be classified into local and global categories according to the granularity of their analysis.

Locally explainable methods aim to help people understand the decision-making process of the learning model for the specific input sample. Some methods are based on back propagation[18, 3, 17], which are usually limited to convolutional neural networks (CNN). Ribeiro et al. introduced Local Interpretable Model Agnostic Explanations (LIME) [8]. Although LIME applies to any model, it assumes that the characteristics of input samples are independent of each other. A game theoretically optimal solution using Shapley values for model explainability was proposed by Lundberg et al. [15], which called SHapley Additive exPlanations (SHAP). This method effectively makes up for the deficiency of LIME by considering the influence of variable groups. Globally explainable methods try to understand the complex logic and the internal working mechanism behind the model as a whole. Classic globally explainable methods include rules extraction [20], Model Distillation [14], Concept Activation Vectors (CAVs) [10] and so on. Furthermore, LIME and SHAP can also provide a global understanding of the model.

4 Method

Fig. 2 shows the overall flow of the method which is divided into two stages. In the first phase, we leverage the historical system call sequences of the data center and use them to train a high-precision intrusion detection model (in Section 4.1). Then, we choose SHAP (in Section 4.2) to obtain global explanations of the trained model and analyze an *important decision interval*. In the second stage, we set a period T to collect real-time system calls in the data center on a regular basis. Then, we can only detect the important fragments through data reduction which is described in Section 4.3.

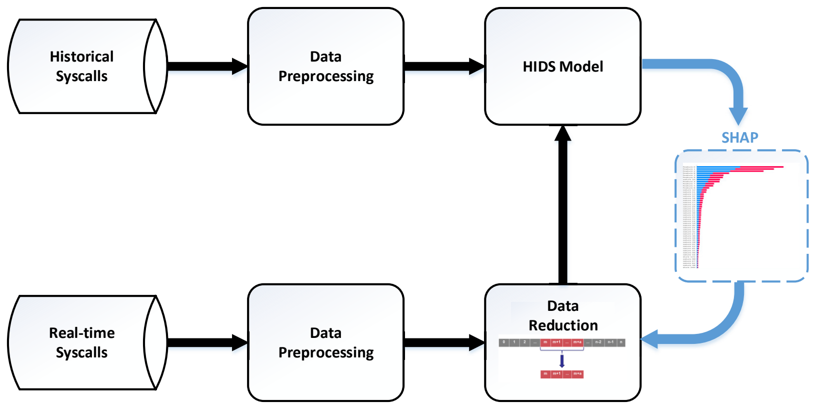


Fig. 2. Overview of the flow of the proposed method

4.1 Intrusion Detection Model

Host-based Intrusion Detection Systems (HIDS) always consider the traces of system calls generated by executing computer programs as input samples. If we treat these traces as sentences, then intrusion detection can be treated as a text classification task. Just like text classification, we need to consider both the local and global features of input sequences during detection. Therefore, our detection model is a combination of two kinds of neural networks: multi-filter convolutional neural network (CNN) and attention-based Bi-directional Long Short-Term Memory (BiLSTM).

Multi-filter CNN Convolutional neural network (CNN) has become one of the most popular neural networks by virtue of its expertise in capturing local dependencies. In CNN, local receptive field is regarded as the input of the bottom layer and the information is transmitted forward through each layer which is composed of filters in order to obtain relevant features of input data. This processing way, which is closer to the visual system of human brain, makes CNN have unique advantages in processing image data. As for 1D data such as system call traces and texts, we can use 1D CNN to generate representations by sliding over sequences.

Nevertheless, traditional CNN only uses one type of convolution kernel for feature extraction which makes it difficult to capture all local dependencies of the entire sequence. In this work, we solve this problem by using multi-filter CNN [11]. We design filters with different sizes in order to get receptive fields of different widths. This allows to extract words of different lengths each time and capture local features at different levels just like n-gram with multi-window size. Therefore, multi-filter CNN can achieve a better extraction effect for the local correlation.

Attention-based BiLSTM In addition to local dependencies, we should also focus on long-range relationships, which is more suitable for RNN. However, RNN reads and updates all previous information, and as the time interval increases, the accumulation of gradients in RNN will approach 0. Some variant of RNN, such as long short-term memory (LSTM) and gated recurrent unit (GRU), have been proposed to solve this problem. Moreover, the improved bidirectional LSTM (BiLSTM) extends the unidirectional LSTM and uses information from the past and the future to learn better.

However, for intrusion detection systems, the sequences of system calls collected in real time are very long (e.g., more than 2000) and important information can appear at any position anywhere in the sequence. So even these variants are difficult to learn reasonable vector representation, which may lead to the decrease of detection accuracy. To tackle these problems, the method which is proposed to do relation classification tasks in the natural language processing domain is transferred to our model [23], which acts the attention mechanism on the output vector produced by the BiLSTM. The representation r of the sequence is actually a weighted sum of the output of the hidden layer in BiLSTM at each time as follows:

$$M = \tanh(H) \tag{1}$$

$$\alpha = softmax(w^T M) \tag{2}$$

$$r = H\alpha^T \tag{3}$$

where:

- H is a matrix consisting of output vectors of the hidden layer.
- w is a trained parameter vector.
- α is the corresponding weights of each output vector of the hidden layer.

In this way, the attention-based BiLSTM can learn different weight coefficient α at different moments by introducing the attention mechanism. This allows to selectively learn the input sequence and capture the long-range dependence relationship better.

Finally, we combine these two neural networks into the intrusion detection model as shown in Fig. 3. Given a trace of system calls, our proposed model generates word-embedding vectors via an embedding layer. Then, we feed these embedding vectors to both two neural networks to create local and global feature vectors. The final vector is the combination of these two vectors. In the end, a fully connected layer is used for classification. In this way, our proposed model can achieve a good detection performance by combining the advantages of multi-filter CNN and attention-based BiLSTM.

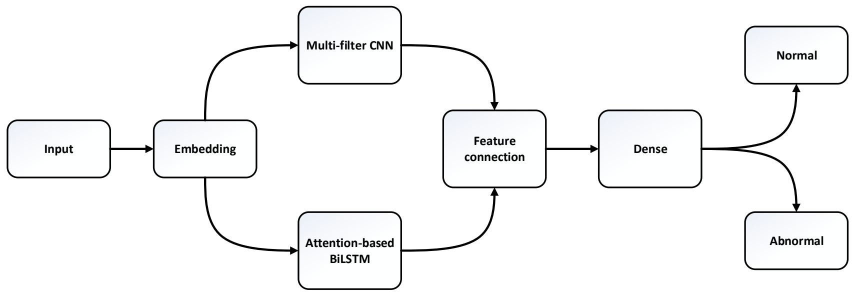


Fig. 3. Overview of the structure of the proposed intrusion detection model

4.2 Using SHAP For Model Analysis

Although our intrusion detection model mentioned above can achieve a good performance, the detection efficiency cannot meet the standards of real-time detection, especially under the circumstance of big data. Therefore, we propose a method to improve detection efficiency by using eXplainable Artificial Intelligence (XAI) to reduce the input data. In the first place, we use SHapley Additive exPlanations (SHAP) to analyze the trained detection model.

SHAP is a general method proposed by [15] to explain model predictions. Inspired by cooperative game theory, SHAP constructed an additive explanatory model that treated all features as “contributors”. For each sample, the model generates a predicted value, and each SHAP value measures how much each feature contributes. SHAP specifies the explanation for an instance x as:

$$y_i = y_{base} + f\left(x_{i1}\right) + f\left(x_{i2}\right) + \ldots + f\left(x_{ik}\right) \tag{4}$$

where:

- y_i is the predicted value of the model.
- y_{base} is the baseline of the entire model (usually the mean value of the target variable of all samples).
- x_i is the ith sample and x_{ij} is the jth feature of it.
- f is the SHAP value. $f(x_i, 1)$ means the contribution of the first feature to the final predicted value y_i in the ith sample. The larger $f(x_i, 1)$ is, the more this feature affects the final decision.

SHAP provides three significant advantages compared to other XAI methods. Firstly, SHAP can work under a black box setting. SHAP does not require any knowledge of the model internals and analyze the model by sending inputs and observing outputs, which effectively support the application environment of intrusion detection. Secondly, SHAP can provide not only explanations for the prediction results of each sample, but also the global explanation for the model, which is what we need. Thirdly, compared with other blackbox methods like LIME, SHAP can better deal with the dependencies between features better by considering the influence of variable groups. In this work, we treat the element at each position in the system call sequence as a feature. Specifically, we label the first system call in the sequence as feature 1, the second as feature 2, and so on. Then, we use SHAP to provide a global explanation of the trained model so that we can obtain the importance of each element to decision-making results of the intrusion detection model.

4.3 Data Reduction

In deep learning, the influence of input features on model is different. Therefore, system calls from different locations have different effects on results of intrusion detection. And because the system call sequence has behavior continuity, there will be some sequence fragments that have little influence on decision-making results of the model. Our aim is to reduce this part of data and retain the fragments that have a large impact on model decisions.

Therefore, after obtaining the global explanation provided by SHAP, we can know which elements in the detection sequence are important for the model and which are not. We select the range of important elements (features) as the *important decision interval* of the input sequence. This allows us to subtract the system call sequences retrieved next based on this interval. In other words, we only use the sequence within the *important decision interval* for model retraining and intrusion detection, so that the efficiency of intrusion detection can be improved on the premise of ensuring the accuracy is not greatly affected. It is worth noting that the length of the *important decision interval s* can be manually set according to the actual situation.

Table 1. The specific composition of datasets

	normal	abnormal	training	detecting	max_len
ADFA-LD	746	746	998	494	3143
UNM(live lpr)	1196	1001	1460	737	7720

5 Evaluation

In this section, we first present the datasets and simulation settings of our experiments. Then, we evaluate the performance of the proposed method and conduct a series of comparisons.

5.1 Datasets and Experimental Configuration

We use two publicly available datasets to conduct our experiments, i.e. ADFA-LD and UNM [4, 2]. Both of them are commonly used in this field. As different types of programs use different mapping files in UNM, we chose the live lpr set which has larger data volume than others. We selected traces with equal proportion in order to balance the normal and abnormal data and divided them into training data and detecting data according to the ratio of 2:1 (see Table 1).

The experimental configuration related to the model operation is described below. The embedding layer transforms one-hot encoding of integers in the call sequence into a dense vector of size 128. Regarding to multi-channel CNN model, we use filter windows of 4, 6, 8, and 10 with 128 feature maps each. The size of memory cells used in BiLSTM stage accounts for 128. We applied the dropout technique with the dropout rate of 0.1 on BiLSTM and the fully connected neural network. In addition, our proposed model is trained by using mini-batch stochastic gradient descent (SGD)for higher accuracy. The size of each mini-batch is 50. And we used Adam optimizers with a learning rate of 0.004. The length of the important decision interval is set to 100. All experiments in this paper are performed under Ubuntu 16.04.6 LTS system, the kernel version is GNU/Linux 3.10.0-1062.12.1.el7.x86_64, the CPU is Intel (R) Xeon (R) with a frequency of 2.40GHz, a NVIDIA TESLA M40 GPU with 24G memory, the version of CUDA is 10.2.

5.2 Experimental Results

Detection accuracy Since most of the traditional methods are relatively old and can achieve low accuracy, we only compare with Support Vector Machines (SVM), which is one of the most common classification methods in machine learning. As shown in Table 2, The precision of our model is 75.6% and 0.2% higher than that of SVM on the two datasets respectively, and the false positives rate (FAR) is reduced by 93.5% and 100%, where a high false-alarm rate can

Table 2. Intrusion detection performance comparison by two evaluation metrics.

Method	ADFA-LD		UNM	
	Precision	FAR	Precision	FAR
SVM	0.557	0.437	0.995	0.00215
Our Proposed	**0.978**	**0.0283**	**0.997**	**0**

Table 3. Ablation experiments

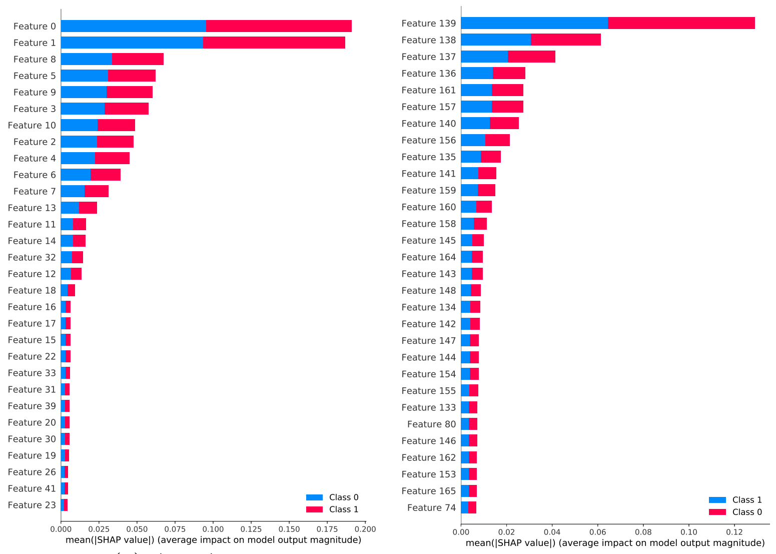
	Multi_filter CNN	BiLSTM	CNN+BiLSTM	Our proposed
ADFA-LD	0.929	0.900	0.965	**0.978**
UNM	0.935	0.983	0.997	**0.997**

affect the performance of HIDS. This proves that our method performs well on detection accuracy.

We also built three other models for Ablation experiments: (1)Multi-filter CNN (2)BiLSTM (3)Multi-filter CNN and BiLSTM. The detailed parameters of these models are same as our proposed model. As shown in Table 3, our model performs best on both two datasets. This result confirms that giving consideration to both the local and global features is better than simply focusing on one type of features. In addition, the accuracy of our model (0.978) is higher than the model with multi-channel CNN and BiLSTM on the ADFA-LD dataset (0.965). As for the UNM dataset, both of their accuracy are reach 0.997. As can be seen from these results, the attention mechanism has a certain improvement in the model effects.

Global Explanation Fig. 4 shows the top 30 important features extracted by SHAP for the ADFA-LD and UNM datasets. Specifically, the y-axis shows features, and the x-axis shows average values of the absolute SHAP values, which reflect average impacts on model output magnitude. Different colors correspond to different classes. The impacts on the Normal and Abnormal classes are marked in blue and red, respectively. The features are ordered according to their importance. Labels of important features extracted for the ADFA-LD dataset are almost less than 100, which means the important pieces of input sequences for the intrusion detection model are in the interval [0,100]. Similarly, as for the UNM dataset, the critical detection segments are between [100, 200]. In this way, we can reduce the input sequences based on these two *important decision intervals*.

Fidelity Tests In order to validate the correctness of the explanation, we design fidelity tests to show whether the features in the selected important decision



(a) ADFA-LD (b) UNM

Fig. 4. The top 30 features extracted by SHAP

Table 4. Fidelity tests by two evaluation metrics.

	ADFA-LD		UNM	
	Precision	FAR	Precision	FAR
x	0.978	0.0283	0.997	0.0
$t(x)_1$	0.862	0.215	0.634	0.0
$t(x)_2$	0.960	0.0769	0.997	0.0

intervals are the major contributors to the detection results. We denote the selected features as F_x. There are two intuitions:

- If features F_x are accurately selected, then removing F_x from input x will greatly affect the detection accuracy.
- If features F_x are accurately selected, only keeping F_x as input has little impact on accuracy.

Using these intuitions, we constructed two new types of input samples $t(x)_1$ and $t(x)_2$ for feature verification. To be more specific, we construct samples $t(x)_1$ by nullifying the selected feature F_x from the original data x and construct $t(x)_2$ by only preserving the feature values of the selected features F_x.

Table 4 shows the results of the fidelity tests. By only nullifying the features of important decision intervals, the accuracy on two datasets decreased by 11% and 36% respectively, and FAR on the ADFA-LD dataset increased from 2%

Table 5. Detection performance before and after data reduction

	ADFA-LD			UNM		
	Precision	Training_time	Detection_time	Precision	Training_time	Detection_time
Before	**0.978**	86.595	2.140	**0.997**	180.350	4.344
After	0.966	**6.306**	**0.232**	0.996	**9.139**	**0.251**
Gain	1%	93%	89%	0.1%	95%	94%

to 21%. This drastic decrease of performance indicates that this small set of features are highly important to the classification. In addition, using only the features in the important decision intervals, the accuracy only decrease by 1% on the ADFA-LD dataset and have no effect on the UNM, indicating that the core patterns have been successfully captured. Therefore, both sets of tests verifies that the features in the important decision intervals we obtained are indeed the major contributors to the detection results.

Before and after data reduction Finally, we compared the results before and after the data reduction. As shown in Table 5, the accuracy of the model is slightly lower than before. However, the training and detection time are greatly reduced. For training duration, the training time of the model on the ADFA-LD dataset is reduced by over 10 times, and nearly 20 times on the UNM dataset. Moreover, the detection time is an order of magnitude lower than beforen and this observation holds for both two datasets. These results confirm that our proposed method can significantly reduce the execution time without affecting the detection accuracy.

In addition, in order to evaluate the detection efficiency in the big data environment, we expand the scale of the dataset. Fig. 5 shows the variation of the detection time before and after data reduction. With the increase of the sample size, the detection time of the method without data reduction grows to a large scale while ours grows slowly, and the gap between them becomes broader and broader.

6 Conclusion

This paper introduces a method to realize high-precision and high-efficiency intrusion detection in the data center. We first design a high-precision intrusion detection model, which can better capture the local and global dependencies of sequence data through integrating multi-filter CNN and attention-based BiLSTM. Next, the important decision interval is proposed through explaining this DNN model by SHAP. Finally, the execution time is shortened by data reduction. The extensive experiments using ADFA-LD and UNM datasets which are public demonstrate that compared with traditional methods, this proposed DNN model can achieve high-precision detection. Furthermore, we also verify that the features in the important decision intervals we obtained are indeed the major

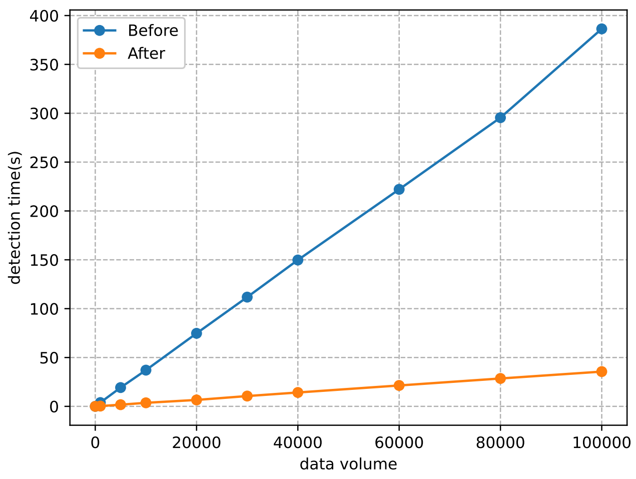


Fig. 5. The variation of detection time before and after data reduction

contributors to the detection results. Moreover, the detection time only produce small increases with the growth of data volume, which verifies the strong adaptability of our method under the environment of big data. Based on the accurate and efficient intrusion detection, our future work is to build a whole set of intrusion detection system and apply it to physical environment.

References

1. Ambusaidi, M., He, X., Nanda, P., Tan, Z.: Building an intrusion detection system using a filter-based feature selection algorithm. IEEE Transactions on Computers (2016)
2. Assem, N., Rachidi, T., Graini, M.T.E.: Intrusion detection using bayesian classifier for arbitrarily long system call sequences. IADIS International Journal on Computer Science and Information Systems **9**(1), 71–81 (2014)
3. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. Plos One **10** (2015)
4. Creech, G., Hu, J.: Generation of a new ids test dataset: Time to retire the kdd collection. In: Wireless Communications and Networking Conference (WCNC), 2013 IEEE (2013)
5. Ding, Yuxin, , , Yuan, Xuebing, , , Zhou, Di, , , and, D.: Feature representation and selection in malicious code detection methods based on static system calls. Computers Security (2011)
6. Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A.: A sense of self for unix processes. In: sp (1996)
7. Ghosh, A.K., Schwartzbard, A., Schatz, M.: Learning program behavior profiles for intrusion detection. In: Workshop on Intrusion Detection and Network Monitoring. vol. 51462, pp. 1–13 (1999)

8. Godsell, M.: Why should we trust you? Marketing (2007)
9. Khreich, W., Murtaza, S.S., Hamou-Lhadj, A., Talhi, C.: Combining heterogeneous anomaly detectors for improved software security. Journal of Systems and Software **137**(MAR.), 415–429 (2017)
10. Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., Sayres, R.: Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav) (2017)
11. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
12. Liao, Y., Vemuri, V.R.: Using text categorization techniques for intrusion detection. In: USENIX Security Symposium. vol. 12, pp. 51–59 (2002)
13. Liu, M., Xue, Z., Xu, X., Zhong, C., Chen, J.: Host-based intrusion detection system with system calls: Review and future trends. Acm Computing Surveys **51**(5), 98.1–98.36 (2019)
14. Liu, X., Wang, X., Matwin, S.: Improving the interpretability of deep neural networks with knowledge distillation. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW) (2018)
15. Lundberg, S., Lee, S.I.: A unified approach to interpreting model predictions (2017)
16. Marteau, P.F.: Sequence covering for efficient host-based intrusion detection. IEEE Transactions on Information Forensics and Security **14**(4), 994–1006 (2018)
17. Shrikumar, A., Greenside, P., Shcherbina, A., Kundaje, A.: Not just a black box: Learning important features through propagating activation differences (2016)
18. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
19. Staudemeyer, R.C.: Applying long short-term memory recurrent neural networks to intrusion detection. South African Computer Journal **56**(1), 136–154 (2015)
20. Tickle, A.B., Andrews, R., Golea, M., Diederich, J.: The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. IEEE Transactions on Neural Networks **9**(6), 1057–1068 (1998)
21. Wang, K., Parekh, J.J., Stolfo, S.J.: Anagram: A content anomaly detector resistant to mimicry attack. In: International Workshop on Recent Advances in Intrusion Detection (2006)
22. Xie, W., Xu, S., Zou, S., Xi, J.: A system-call behavior language system for malware detection using a sensitivity-based lstm model. In: Proceedings of the 2020 3rd International Conference on Computer Science and Software Engineering. pp. 112–118 (2020)
23. Zhou, P., Shi, W., Tian, J., Qi, Z., Xu, B.: Attention-based bidirectional long short-term memory networks for relation classification. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (2016)