Predicting severity of software vulnerability based on BERT-CNN

Xuming Ni, Jianxin Zheng, Yu Guo, Xu Jin, Ling Li
Information & Telecommunication Branch
State Grid Jinhua Power Supply Company
Jinhua, China
nixuming@126.com, zheng_jianxin@zj.sgcc.com.cn, 77979292@qq.com, jinqqxu@qq.com, pujiangliling@qq.com

2022 International Conference on Computer Engineering and Artificial Intelligence (ICCEAI) | 978-1-6654-6803-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCEAI55464.2022.00151

Abstract — **Software vulnerabilities threaten the security of computer system, and recently more and more loopholes have been discovered and disclosed. For the detected vulnerabilities, the relevant personnel will analyze the vulnerability characteristics, and combine the vulnerability scoring system to determine their severity level, so as to determine which vulnerabilities need to be dealt with first. In recent years, some characteristic description-based methods have been used to predict the severity level of vulnerability. However, the traditional text processing methods only grasp the superficial meaning of the text and ignore the important contextual information in the text. Therefore, this paper proposes an innovative method, called BERT-CNN, which combines the specific task layer of Bert with CNN to capture important contextual information in the text. First, we use Bert to process the vulnerability description and other information, including Access Gained, Attack Origin and Authentication Required, to generate the feature vectors. Then these feature vectors of vulnerabilities and their severity levels are input into a CNN network, and the parameters of the CNN are gotten. Next, the fine-tuned Bert and the trained CNN are used to predict the severity level of a vulnerability. The results show that our method outperforms the state-of-the-art method with 91.31% on F1-score.**

Keywords- Bert; CNN; software vulnerability; vulnerability severity prediction

I. INTRODUCTION

With the rapid development of computer technology, the use of all kinds of software brings convenience to our life. However, it is inevitable that there are some loopholes in software, which threaten the security of computer system. Thousands of vulnerabilities are discovered and disclosed every year. Software security experts will analyze these vulnerabilities and usually use the Common Vulnerability Scoring System (CVSS) [1] to evaluate the them. In the face of so many vulnerabilities, software suppliers and security experts can not repair all of them in time. Therefore, they need to pay more attention to those vulnerabilities with high severity level and solved them first. Generally speaking, severity is an important characteristic of vulnerability, which indicates the harm degree of software. For each vulnerability, security experts used to score the vulnerability through a series of complex calculations based on CVSS.

The existing vulnerability severity prediction methods are mainly divided into two categories: machine learning-based and deep learning-based. The methods based on machine learning mainly pre-process vulnerability data and make prediction by establishing relevant features and combining them with the machine learning model. The performance of this method largely depends on feature engineering. As the number of public vulnerabilities continues to increase, and the scale of the vulnerability data also grows larger, so that the feature engineering will cost a lot of manpower and time. On the contrary, the methods based on deep learning use vulnerability descriptions to predict the severity level, which is more efficient than the machine learning-based methods. However, the deep learning-based methods only use the vulnerability description to predict the severity level, while ignore other information that is helpful for severity prediction.

To address the above issues and improve the prediction effect, we propose an innovative method, called BERT-CNN, which combines the specific task layer of Bert with CNN to capture important contextual information in the text. First, Bert is adopted to generate the feature vectors of software vulnerabilities. Different from state-of-the-art methods, we use not only the vulnerability description from NVD but also other information from CVE Details, including Access Gained, Attack Origin, Authentication Required. Then these feature vectors of vulnerabilities and their severity level are input into a CNN network, and the parameters of the CNN are gotten. Next, the fine-tuned Bert and the trained CNN are used to predict the severity level of vulnerability. The main contributions of this paper are as follows:

- We proposed an innovative method for the severity level prediction, which combines Bert and CNN.
- In our model, not only the vulnerability description from NVD but also other information from CVE details are adopted to get more detailed features.

The rest of this paper is organized as follows. In Section II, we review the related work of software vulnerability research in recent years. In Section III, our BERT-CNN model is described in detail. In Section IV some experiments are taken to evaluate the effectiveness of our method. Section V draws the conclusion.

II. RELATED WORK

The research of software vulnerability is a hot topic in the field of software security. Researchers have studied software vulnerabilities from different aspects. Huang et al. [2] proposed an automatic vulnerability classification model, TFI-DNN, which works better on vulnerability category prediction compared with traditional SVM, Bayes models, etc. Shuai et al. [3] improved the SVM classifier to predict

978-1-6654-6803-9/22/$31.00 ©2022 IEEE
DOI 10.1109/ICCEAI55464.2022.00151

the category of vulnerabilities by analyzing the National Vulnerability Database (NVD [4]), and achieved good results. The sharp increase in the number of vulnerabilities has brought great challenges to the vulnerability analysis of software security experts. Jeona et al. [5] proposed the Software Vulnerability Analysis System (AutoVAS). They use deep learning techniques to express the source code as embedding vectors to analyze vulnerabilities. They have used the AutoVAS system to find three zero-day vulnerabilities. Harer et al. [6] proposed a vulnerability detection model based on machine learning for C and C + + programs, and concluded through comparative experiments, which show that the combination of CNN and tree based model can achieve better results. Exploitability is one of the important features of software vulnerability, which is a topic of concern. Yin et al. [7] proposed the prediction model ExBERT by using transfer learning, which predicts the exploitability through vulnerability description. Lyu et al. [8] combined the vulnerability description and characteristics, extracted finer granularity character-level features through charCNN to predict the exploitability of vulnerability. The effect of this method is slightly improved than ExBERT.

For the study of vulnerability's severity level prediction, Sharma et al. [9] used word embedding and convolutional neural network to predict the severity level of vulnerability. They classified vulnerability's severity level into three categories (Low, Medium, High). Han et al. [10] used the word embedding and a one-layer shallow CNN method to predict severity level through vulnerability description, they classified the severity level into four categories (Low, Medium, High, Critical). Their method only uses the vulnerability description to predict the vulnerability's severity level, while ignoring other vulnerability information. Our method is different from the above methods. We use not only the vulnerability description from NVD but also other information from the CVE Details [11], including Access Gained, Attack Origin, Authentication Required. And then we adopt Bert to generate more detailed feature vectors.

III. METHODOLOGY

The severity level of vulnerability is critical to determine the priority of vulnerability's repair, and those high severity vulnerabilities need to be solved first. Therefore, we propose an innovative method, called BERT-CNN, which combines the specific task layer of Bert with CNN, to predict the vulnerability's severity level. The model extracts feature from vulnerability description and other relevant information, including Access Gained, Attack Origin and Authentication Required, to predict the severity level through a multi-class text classification task.

A. Overview of Method

The main workflow of our method is shown in Fig. 1. Firstly, we process the vulnerability data collected from NVD and CVE Details to get the vulnerability description and other relevant vulnerability information. We choose the severity level information to mark the vulnerability. The labels corresponding to the four severity levels (Low, Medium, High, Critical) of the vulnerability are 0,1,2,3 respectively. Then we input the vulnerability description and other relevant vulnerability information (Access Gained, Attack Origin and Authentication Required) into the Bert layer for text vectorization. Next, we process the outputs of Bert layer and labels of severity levels to obtain the feature vectors of the vulnerability. The feature vectors are extracted through the convolution and pooling layer of CNN, and finally the classification results are obtained through the fully connected layer. The details of some key sections in our method are described as follows.

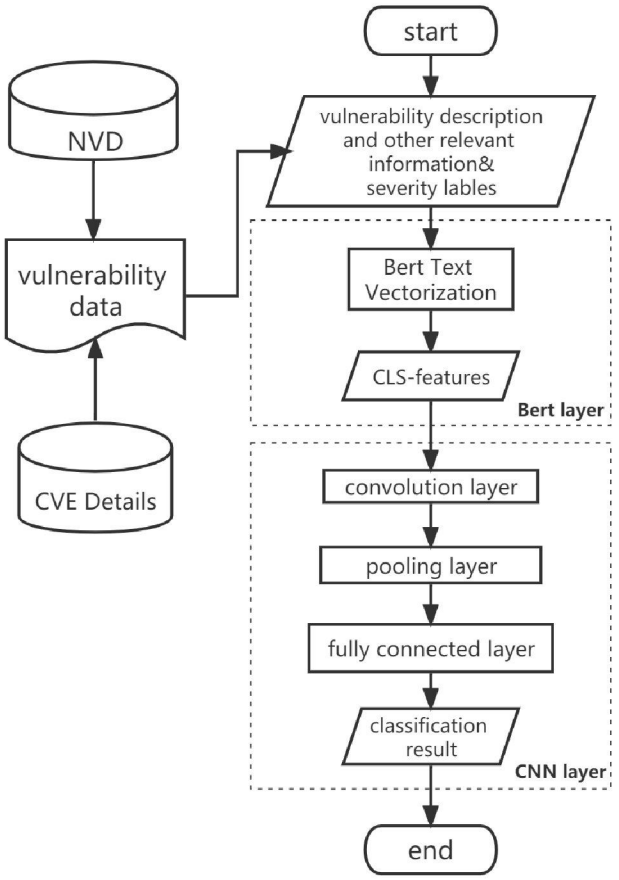

Figure 1. Model structure of BERT-CNN.

B. CLS-features Input

Bert [12] was proposed by J. Devlin et al. of Google AI language, which achieved most advanced results on multiple NLP tasks, causing a stir in the field of deep learning. The input of Bert consists of three parts: token embedding, segment embedding and position embedding, as shown in Fig. 2.

The main structure of Bert is a transformer, which uses the encoder part of the transformer, and the encoder stack of multiple layer transformer forms the overall framework of Bert. We input the merged embedding into the encoder layer. In the encoder, self-attention is an important part. It can help the current node not only focus on the current word, but also obtain the semantic information of the context. For example,

for the following vulnerability description "The affected product is vulnerable to a heap-based buffer overflow, which may lead to code execution.", we humans can easily judge what the word "which" means here, but it is difficult for machines to judge, while self-attention can help the machine to connect the word "which" with "buffer overflow". Self-attention can learn the context semantics of vulnerability information, so as to better capture vulnerability features.

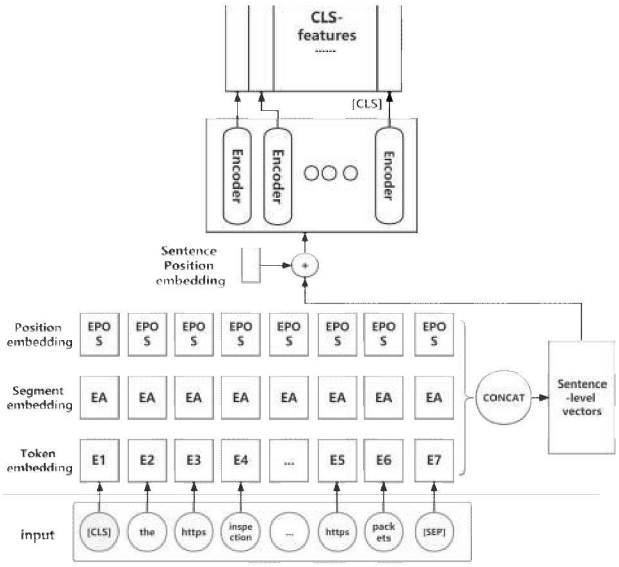

Figure 2. BERT layer.

The information captured by different layers of Bert is different, the lower layers are biased towards syntactic feature learning, and the higher layers are biased towards semantic feature learning. In order to better predict the vulnerability's severity level, we need to obtain the deeper semantic features of the vulnerability information. For the Bert model, the text classification task needs to focus on the output of CLS, so our model extracts the CLS vectors from layer 1 to 12 and splice them together. We name it CLS-features, and use this as the input of CNN.

C. *Feature extraction and classification*

The CLS-features of each vulnerability and its label are input into a CNN to train the parameters of the CNN. Then the trained CNN is adopted to predict the severity level of vulnerabilities. The structure of the CNN is shown in Fig. 3.

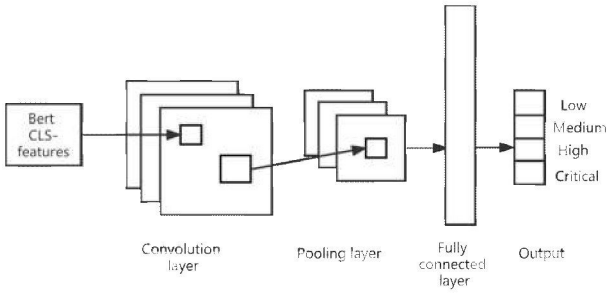

Figure 3. CNN layer.

The input is a matrix representing the sentence that we need to extract the features through the convolution and pooling layers of the CNN. In convolution layer, different n-gram features are extracted by convolution. Suppose the matrix dimension we input is $m*n$, m expresses that each sentence has m words and n expresses each word is represented by a word vector of n dimensions. We take the words as the minimum granularity of the text during the feature extraction, so we set the width of the convolution kernel C to n and the height to h. The convolution kernel C is convoluted with h words, and it is processed by the activation function to obtain the corresponding feature T_i. We use $S_{i:i+h-1}$ to express these h words from S_i to S_{i+h-1}. The process is shown in

$$T_i = f(C \bullet S_{i:i+h-1} + b) \quad (1)$$

where f is the activation function, b is the offset we added.

A $m-h+1$ dimensional vector V is obtained after the convolution operation shown in

$$V = [v_1, v_2, ..., v_{m-h+1}] \quad (2)$$

In the convolution process, due to the different heights of convolution kernels, the vector dimensions obtained after convolution are inconsistent, so we need to use the pooling strategy. For each feature vector, we use 1-max pooling to extract the maximum value that represents it in the pooling layer, which guarantees the capture of important features. In order to obtain the final feature vectors of the pooling layer, the values obtained by 1-max pooling need to be spliced in the fully connected layer. Finally, we get classification result from the fully connected layer. For example, for the following vulnerability "The affected product is vulnerable to a heap-based buffer overflow, which may lead to code execution.", we can get the classification result that the severity level of this vulnerability is "Medium".

In this paper, we used the ReLu [13] as an activation function, which enhances the nonlinear characteristics of the network, makes the network more robust and can learn complex data. Moreover, it can generate nonlinear mapping between input and output. The activation function can be expressed in

$$f(x) = \max(0, x) \quad (3)$$

When the input value is less than 0, the function value is equal to 0; when the input value is greater than 0, the function value is equal to the input value.

IV. EXPERIMENT

Several experiments were taken to evaluate the proposed BERT-CNN model, which was implemented with Python in PyCharm 2019.3.1 x64 on the Google Colab under Linux system, all experiments run with PyTorch on an NVIDIA Tesla K80 GPU.

A. *Data Set*

In this study, we used datasets from the NVD and CVE Detials. We collected software vulnerabilities from 1999 to 2020, which contained 149,773 pieces of data. Each data

contains the vulnerability description and other relevant information, as shown in Fig.4. For the prediction study of the vulnerability's severity level, we randomly selected 32,000 pieces of data and divided them into four categories (Low, Medium, High, Critical) according to their severity level. The proportion of training set, validation set and test set is 80%, 10% and 10% respectively.

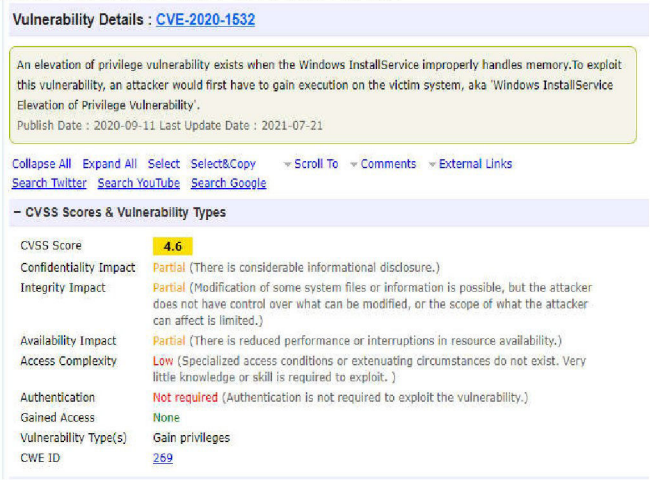

Figure 4. Example of software vulnerability information

B. Baseline Settings

To illustrate the effect of our method, we set four baselines. In order to prove the BERT-CNN model performs better in predicting vulnerability's severity level, we design Bert and CNN separately to compare the effect with BERT-CNN. In conclusion, we have designed the following two models: word2vec+CNN and Bert.

1) word2vec+CNN: We use word2vec as the pre-trained model of word vector. In the convolution layer, in order to extract phrase features from the input text description, we set the convolution core size to (2,3,4) and the number of convolution cores in each group to 256. After the convolution operation, the generated feature map was processed with a maximum pooling strategy. We used ReLu as the activation function of the convolution layer. About the model parameters, the learning rate is 1e-3, batch size is 64, maximum text length L is 125, and epoch is 30.

2) Bert: We use the pre-trained model "bert-base-uncased" and fine-tune Bert through the vulnerability information from NVD and CVE Details. The learning rate of the Bert model is 5e-5, batch size is 64, maximum text length L is 128, and epoch is 20.

3) In addition, we also compare our method with the state-of-the-art methods, including GloVe+CNN [9] and word2vec+CNN [10]. These two methods predict the severity level using only vulnerability description.

C. Evaluation Metrics

The classification results are usually expressed as a $z \times z$ confusion matrix, and z represents the number of classes. Each column of the confusion matrix represents the predicted class while each row represents the actual class. In our experiment, four severity levels are adopted for vulnerabilities, so the value of z is 4. This paper uses the evaluation indicators used in the literature [14, 15] to evaluate the prediction effectiveness: Accuracy, Precision, Recall, and F1-score. In the evaluation metrics, TP is the number of samples classified as positive, which are actually positive samples; FN is the number of samples classified as negative, which are actually positive samples; FP is the number of samples classified as positive, which are actually negative samples; TN is the number of samples classified as negative, which are actually negative samples.

Accuracy: the ratio of correctly classified samples to the total number of samples, as in

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

Precision: it shows the probability of actually positive samples among all the samples predicted to be positive, as in

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Recall: it shows the probability of the samples predicted to be positive among all the samples that are actually positive, as in

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

F1-score: it is a comprehensive evaluation metric of precision and recall, which shows the harmonic mean of the precision and recall values, as in

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7}$$

D. Experimental Results and Analysis

We use the 10-fold cross validation method for experiments. The data set are divided into 10 equal parts, we chose one as the test set, and the rest are used as training data and development data in turn. In order to verify whether our model is superior to the baseline models, we do comparative experiments under the same data set. The results are presented in Table I. We can see that the classification effect of BERT-CNN is better than other models we designed, which can achieve 91.38% on the accuracy, 91.36% on precision, 91.30% on recall and 91.31% on F1-score. Compared with Bert and word2vec+CNN, our method improves F1 score by 1.44% and 3.2% respectively. In addition, compared with the existing methods GloVe+CNN [9] and word2vec+CNN [10], the F1 value of our method has an improvement 16.13% and 18.06% respectively, which proves that our method performs better than predicting the severity level using only vulnerability description.

In order to fully verify the effectiveness of our method, we also evaluate the prediction effect of each severity level. The confusion matrix for our method is shown in Table II and the results for each class are presented in Table III. We can see that our method performs the best in all the metrics for all severity levels, except the precision for the low level where the Baseline2 is 0.008 higher than our method.

TABLE I. THE EXPERIMENTAL RESULTS

TABLE II. CONFUSION MATRIX

TABLE III. THE RESULTS FOR EACH CLASS

	accuracy	precision	recall	F1-score
Sharma et al. [9]	75.87%	75.82%	75.58%	75.18%
Han et al. [10]	73.71%	73.40%	73.39%	73.25%
Baseline1 word2vec+CNN	88.06%	88.24%	88.04%	88.11%
Baseline2 Bert	89.94%	89.93%	89.89%	89.87%
Our method BERT-CNN	91.38%	91.36%	91.30%	91.31%

labels	Low	Medium	High	Critical
Low	**730**	48	0	9
Medium	43	**691**	51	17
High	7	50	**692**	31
Critical	6	12	2	**811**

		Low	Medium	High	Critical
Sharma et al. [9]	precision	83.69%	77.60%	65.77%	76.20%
	recall	90.61%	56.79%	69.43%	85.47%
	F1-score	87.02%	65.58%	67.55%	80.57%
Han et al. [10]	precision	80.86%	67.95%	68.64%	76.34%
	recall	82.23%	73.08%	57.89%	80.37%
	F1-score	81.45%	70.42%	62.81%	78.30%
Baseline1 word2vec+ CNN	precision	92.75%	80.33%	91.75%	88.13%
	recall	90.93%	83.31%	86.62%	91.30%
	F1-score	91.83%	81.80%	89.11%	89.69%
Baseline2 Bert	precision	93.68%	81.67%	91.75%	92.62%
	recall	91.64%	85.85%	86.22%	95.87%
	F1-score	92.65%	83.71%	88.90%	94.22%
Our method BERT-CNN	precision	92.88%	86.27%	92.89%	93.43%
	recall	92.76%	86.16%	88.72%	97.57%
	F1-score	92.82%	86.21%	90.75%	95.47%

V. CONCLUSION

In this paper, we propose an innovative method, called BERT-CNN, which combines the specific task layer of Bert with CNN to capture important contextual information in the text. First, Bert is adopted to generate the feature vectors of software vulnerabilities. Different from state-of-the-art methods, we use not only the vulnerability description from NVD but also other information from CVE details, including Access Gained, Attack Origin, Authentication Required. Then these feature vectors of vulnerabilities and their severity levels are input into a CNN network, and the parameters of the CNN are gotten. Next, the fine-tuned Bert and the trained CNN are used to predict the severity level of a vulnerability according to its descriptions. The results show that our method outperforms the state-of-the-art method with 91.31% on F1-score.

ACKNOWLEDGMENT

This work has partially been sponsored by the State Grid Zhejiang Electric Power Co. LTD. Science and Technology project (Grant No. 5211JH2000RZ).

REFERENCES

[1] FIRST, Common vulnerability scoring system (cvss) version 2.0, https: //www.fifirst.org/cvss/v2/guide#i1.2, 2007, [Online; accessed 15-March- 2022].

[2] G. Huang, Y. Li, and Q. Wang, Automatic Classification Method for Software Vulnerability Based on Deep Neural Network, IEEE Access, 2019, 7:28291-28298.

[3] B. Shuai, H. Li, M. Li, Q. Zhang, and C. Tang, Automatic classification for vulnerability based on machine learning, IEEE International Conference on Information and Automation, 2013:312–318.

[4] NVD, National vulnerability database home, https://nvd.nist.gov/, [Online; accessed 15-Nov- 2021].

[5] S. Jeona and H. K. Kim, AutoVAS: An automated vulnerability analysis system with a deep learning method, Computers & Security, 2021, 106:102308.

[6] J. A. Harer et al., Automated software vulnerability detection with machine learning, 2018, arXiv:1803.04497. [Online]. Available: http://arxiv.org/abs/1803.04497.

[7] J. Yin, M. Tang, J. Cao, and H. Wang, Apply transfer learning to cybersecurity: Predicting exploitability of vulnerabilities by description, Knowledge-Based Systems, 2020, 210:106529.

[8] J. Lyu, Y. Bai, Z. Xing, X. Li, and W. Ge, A Character-Level Convolutional Neural Network for Predicting Exploitability of Vulnerability, International Symposium on Theoretical Aspects of Software Engineering, 2021: 119-126.

[9] R. Sharma, et al., Software vulnerability prioritization using vulnerability description, International Journal of System Assurance Engineering and Management, 2021, 12(1): 58-64.

[10] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, Learning to Predict Severity of Software Vulnerability Using Only Vulnerability Description, IEEE International Conference on Software Maintenance and Evolution, 2017: 125-136.

[11] CVE Details, https://www.cvedetails.com/, [Online; accessed 15-March- 2022].

[12] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019, 1:4171–4186.

[13] Ç. Gulçehre, M. Moczulski, M. Denil, and Y. Bengio, Noisy Activation Functions, 33rd International Conference on Machine Learning, 2016, 48: 3059-3068.

[14] H. Kekul , B. Ergen, and H. Arslan, A multiclass hybrid method to est imating software vulnerability vectors and severity score, Journal of I nformation Security and Applications, 2021, 63:102038.

[15] S. Nakagawa, T. Nagai, H. Kanehara et al., Character-Level Convolutional Neural Network for Predicting, IEICE Transactions on Information and Systems, 2019, E102.D(9):1679-1682.