Machine Learning: Research on Detection of Network Security Vulnerabilities by Extracting and Matching Features

Ying Xue

Shaanxi Police College, Xi'an, Shaanxi 710021, China
E-mail: yhn3br@163.com

Received 20 March 2023; Accepted 17 May 2023;
Publication 12 August 2023

Abstract

The existence of vulnerabilities is a serious threat to the security of networks, which needs to be detected timely. In this paper, machine learning methods were mainly studied. Firstly, network security vulnerabilities were briefly introduced, and then a Convolutional Neural Network (CNN) + Long Short-Term Memory (LSTM) method was designed to extract and match vulnerability features by preprocessing vulnerability data based on National Vulnerability Database. It was found that the CNN-LSTM method had high training accuracy, and its recall rate, precision, F1, and Mathews correlation coefficient (MCC) values were better than those of support vector machine and other methods in detecting the test set; its F1 and MCC values reached 0.8807 and 0.9738, respectively; the F1 value was above 0.85 in detecting different categories of vulnerabilities. The results demonstrate the reliability of the CNN-LSTM method for vulnerability detection. The CNN-LSTM method can be applied to real networks.

Keywords: Machine learning, network security, vulnerability detection, extracting features, convolutional neural network, matching features.

Journal of Cyber Security and Mobility, Vol. 12_5, 697–710.
doi: 10.13052/jcsm2245-1439.1254

1 Introduction

With the rapid development of network technology, network security faces increasingly serious challenges under the influence of network timeliness and openness [1], and network security is also gaining widespread attention [2]. Network security vulnerabilities are widespread in systems and software [3], and attackers use vulnerabilities to steal and manipulate data, jeopardizing the normal operation of the system while also seriously threatening the security of network data [4], so it is increasingly important to detect vulnerabilities in timely and efficiently. The detection of vulnerabilities is conducive to the timely repair of vulnerabilities to avoid major problems in the system, thus ensuring system security [5]. However, with the increase of vulnerabilities, the traditional vulnerability detection methods have lagged behind [6]. Advances in machine learning technology have provided many new methods for vulnerability detection [7]. Xiao et al. [8] designed a binary-level vulnerability detection technique and found through experiments that the method achieved 100% accuracy for one-day vulnerabilities and 87.6% accuracy for recurring vulnerabilities. Based on the theory of automata, Minaev et al. [9] developed a structural model for unstable network interactions and found through experiments that the model could detect vulnerabilities in the network in a timely and complete manner. Nancy et al. [10] designed a decision tree algorithm based on intelligent fuzzy temporal state and found that the method had a good detection effect, with low false alarm rate, energy consumption, and delay through experiments. For buffer overflow vulnerability, Ren et al. [11] designed a method based on software metric and decision tree algorithm and found that the method achieved 82.53% and 87.51% accuracy in detecting different data sets. Kovtun et al. [12] proposed a mathematical tool to simulate the process of operating an information system in an aggressive cyber environment, taking into account the cumulative parameters of functional efficiency of the studied system, operational risks, and the amount of resources invested in cybersecurity measures during the design phase. Alqarni et al. [13] developed a method to detect source code vulnerabilities in the software development process, which solved the data imbalance problem through the undersampling technique. The experiment showed that the accuracy of the method was 96.46% for the SATE IV Juliet dataset and 77.16% for the balanced dataset of github and debian. Chakraborty et al. [14] designed an automated vulnerability detection method based on deep learning, which improved the accuracy by 33.57% and recall rate by 128.38% compared to the baseline model. Munonye et al. [15] conducted a study to detect

vulnerabilities in OAuth authentication and authorization processes and used a machine learning-based approach. It was tested and found to have over 90% accuracy. To address the shortcomings of existing methods in vulnerability detection, in order to further improve the accuracy of vulnerability detection and the adaptability of the algorithm in the face of complex vulnerabilities, this paper developed the Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) neural network methods, providing a theoretical support to solve the delay and complexity of vulnerability detection and helping to reduce the risk of network intrusion and establish a harmonious network environment. The organization structure of this paper is as follows. Section 1 introduces the concept of network security vulnerability and the current trend of vulnerability and analyzes the pre-processing method of vulnerability features. Section 2 describes the vulnerability feature extraction and matching algorithm based on CNN+LSTM, which extracts vulnerability features in CNN layer and LSTM layer, and then performs matching in softmax layer to realize the classification of vulnerability. Section 3 conducts experiments on the method proposed in this paper to prove the performance of the method in vulnerability detection by analyzing the results. Section 4 gives a brief summary of the paper and gives the limitations of the research and the direction of future research.

2 Network Security Vulnerability Overview and Data Pre-processing

A vulnerability is a flaw in the hardware or software of a system that does not affect the security of the system but is a condition for an attacker to take control of the system, steal or manipulate data [16].

The existence of vulnerabilities may be related to the following factors: (1) improper deployment by administrators; (2) defects in network protocols; (3) defects in software design; and (4) defects in business interaction design.

According to the China National Vulnerability Database of Information Security (CNNVD) (https://www.cnnvd.org.cn/home/report), a total of 199,523 vulnerabilities have been collected as of December 31, 2022. According to CNNVD statistics, the number of new vulnerabilities per month from July 2022 to December 2022 is shown in Figure 1.

From Figure 1, it was seen that the number of vulnerabilities was always in a faster growth, and the average number of new vulnerabilities reached 2066 per month in the past six months, indicating that the vulnerability

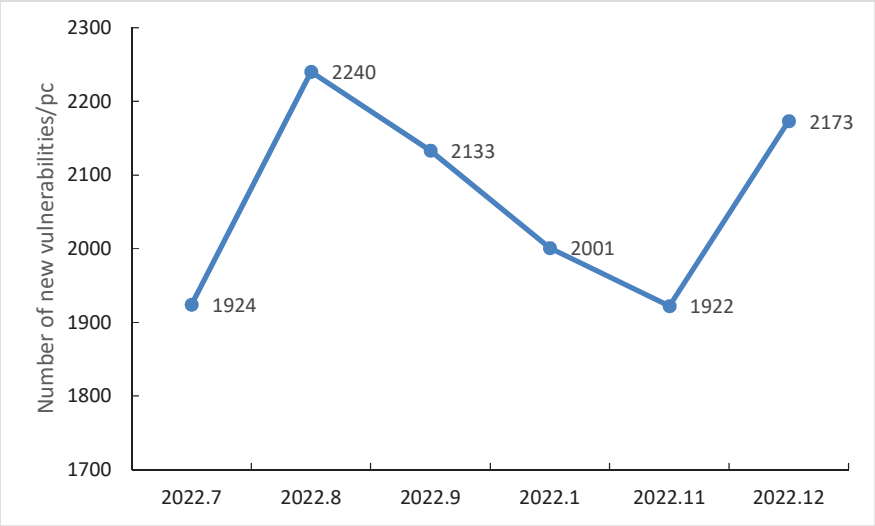

Figure 1 Number of new vulnerabilities per month from July 2022 to December 2022 (Source: https://www.cnnvd.org.cn/home/report).

Table 1 Top 10 types of vulnerabilities in December 2022

Vulnerability Type	Number of Vulnerabilities/n	Percentage
Cross-site scripting	350	16.11%
Buffer error	248	11.41%
Structured query language injection	104	4.79%
Code issues	98	4.51%
Input validation error	73	3.36%
Path traversal	58	2.67%
Resource management errors	58	2.67%
Cross-site request forgery	46	2.12%
Authorization issues	43	1.98%
Command injection	31	1.43%

situation is relatively serious and the demand for vulnerability detection is high.

The top 10 vulnerability types with high percentages in December 2022 are shown in Table 1.

From Table 1, it was seen that the type of vulnerability that accounted for the largest percentage was cross-site scripting (XSS). This type of vulnerability means that the attacker modifies the user's Uniform Resource Locator

(URL), injects malicious code, and then attacks the users when they browse the page, and it is one of the most common vulnerabilities [17].

In vulnerability detection, in order to be able to facilitate the recording and analysis of vulnerabilities, it is necessary to describe the vulnerability. Vulnerability description refers to the description of the cause, location, and other information of the vulnerability according to certain specifications. Common Vulnerabilities & Exposures (CVE) provides the same description of the vulnerability and uses the CVEID to distinguish between different vulnerabilities. The National Vulnerability Database (NVD) is a free and open database [18] that stores more than 170,000 vulnerability information since 2000. The CVEID is represented by "CVE-2021-0001", which means "CVE-year-number", and the original data is presented in English. In this paper, the vulnerability features were extracted based on the NVD. First, the CVEID, vulnerability category, and vulnerability description were obtained from the NVD, and then the vulnerability features were extracted to match the features with the categories.

For the original English text, the original English text is first cleaned by word segmentation, removal of non-text symbols, and removal of stop words, and then, two methods, Term Frequency-Inverse Document Frequency (TF-IDF) and mutual information (MI) [19], are used to preprocess the cleaned text.

TF-IDF refers to term frequency (TF) and inverse document frequency (IDF) [20]. For preprocessed corpus $D_{n\times 3}$ (vulnerability ID, vulnerability category, vulnerability description), firstly, the TF-IDF value of feature word $w_{i,j}$ is calculated: $TF - IDF(w_{i,j}, d_i) = TF(w_{i,j}, d_i) \times IDF(w_{i,j})$; then, the TF-IDF value is converted into two-dimensional matrix $T_{n\times W}$ by one-hot encoding, where $W = \sum_{i=1}^{n} size(d_i)$, $1 \leq i \leq n$. Each column of the matrix refers to the TF-IDF value of feature word $w_{i,j}$ in n texts.

Then, based on two-dimensional matrix $T_{n\times W}$ and the category labels of the vulnerabilities, the MI value of each feature word and its category label is calculated:

$$MI(w_{i,j}, c) = \sum_{w_{i,j}\in T} \sum_{c\in C} p(w_{i,j}, c) \log_2 \frac{p(w_{i,j}, c)}{p(w_{i,j})p(c)} \tag{1}$$

where T refers to the set of all feature words, $p(w_{i,j}, c)$ refers to the occurrence frequency of $w_{i,j}$ belonging to category c, and $p(w_{i,j})$ and $p(c)$ are the occurrence frequencies of $w_{i,j}$ and category c, respectively.

Threshold value $\vartheta(\vartheta \in (0, 1))$ is set, and k keywords with $MI_k > \vartheta$ are screened.

The Skip-gram model in Word2Vec [21] is used to learn corpus $D_{n\times 3}$ to obtain word vector $v(w)$; then, $MI(w)$ obtained after screening is indexed and weighted with $v(w)$ to obtain final word vector $V(w) = v(w) \cdot e^{MI(w)}$.

3 Vulnerability Feature Extraction and Matching Algorithm based on CNN and LSTM

The CNN is a very widely used model in machine learning [22], which can achieve better feature extraction through convolutional and pooling layers. It can effectively extract useful features and eliminate irrelevant features while having faster training speed, and it has good performance in image processing [23], speech processing [24], and so on. LSTM can hold sequence information for a long time and process information, which has a wide range of applications in speech recognition [25], data prediction [26] and other fields. This paper combined the CNN with LSTM and used the CNN+LSTM hybrid model to achieve the detection of vulnerabilities. The specific process of the algorithm (Figure 2) is as follows.

(1) The word vector obtained in the previous section was used as the input of the CNN layer. The vulnerability features were extracted by the convolutional and pooling layers.

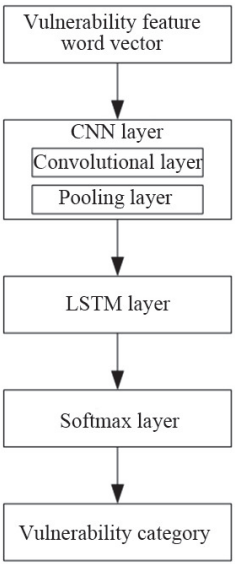

Figure 2 The CNN+LSTM hybrid model.

(2) The new feature vector obtained from the CNN layer was used as the input of the LSTM layer to further extract features.
(3) The feature results obtained from the LSTM layer were classified in the softmax layer, and the vulnerability category was output.

In the CNN layer, for word vector $V(w)$:$V(w)_i = (x_0, x_1, \ldots, x_n)$, let the sliding window be h, and the new window vector is written as: $X_k = [v(w)_k, v(w)_{k+1}, \cdots, v(w)_{k+h-1}]$. By convolution kernel C, the current window feature vector is calculated: $Y_k = f(X_k \otimes C + b)$, where b is the bias and f is the activation function, $f = \frac{1}{1+e^{-x}}$. After the convolution, the most important features are extracted by maximum pooling in the pooling layer: $B_k = MAX(Y_{k-h+1}, Y_{k-h+2}, \ldots, Y_k)$.

In the LSTM layer, B_k obtained by the CNN layer is taken as the input. In the LSTM layer, the processing of the features is done by the updates of the forgetting gate, the input gate, and the output gate. The update of the output of the forgetting gate is written as: $f_t = \sigma(w_f h_{t-1} + u_f x_t + b_f)$, the update of the input gate output is written as: $i_t = \sigma(w_i h_{t-1} + u_i x_t + b_i)$, $\tilde{C}_t = \tanh(w_c h_{t-1} + u_c x_t + b_c)$, the update of the cell state is written as: $C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$, and the update of the output of the output gate is written as: $o_t = \sigma(w_o h_{t-1} + u_o x_t + b_o)$, $h_t = O_t \times \tanh(C_t)$. In these equations, σ stands for the sigmoid function, h_t refers to the hidden state at moment t, $\mathrm{x_t}$ is the input vector, $\mathrm{w_f}$, $\mathrm{w_i}$, $\mathrm{w_c}$, and $\mathrm{w_o}$ are the weights of different gates of $\mathrm{h_t}$, $\mathrm{u_f}$, $\mathrm{u_i}$, $\mathrm{u_c}$, and $\mathrm{u_o}$ are the weights of different gates of $\mathrm{x_t}$, and $\mathrm{b_f}$, $\mathrm{b_i}$, $\mathrm{b_c}$, and $\mathrm{b_o}$ are the biases of different gates.

Ultimately, the output of the LSTM is classified in the softmax layer, and the formula is written as:

$$P(y = i|x, \theta) = \frac{e^{o_i}}{\sum_{k=1}^{N} e^{o_k}}, \tag{2}$$

where $\mathrm{P(y = i|x}, \theta)$ refers to the probability of sample x belonging to category i. The result of vulnerability detection is obtained by matching the feature results of the LSTM layer to the category with the highest probability.

4 Experiment and Analysis

4.1 Experimental Setup

The experimental environment was a TensorFlow/Keras deep learning framework based on Python 3.7, Windows 10 operating system, Intel(R) Core(TM)i7-8700 CPU, and 16 GB of random access memory. The batch size

Table 2 Experimental vulnerability situation

Vulnerability ID	Vulnerability Category	Category Label
CWE-79	Cross-site scripting	0
CWE-119	Buffer error	1
CWE-20	Input verification	2
CWE-200	Information leakage	3
CWE-787	Out-of-bound write	4

Table 3 Confusion matrix

Confusion Matrix		True Value	
		Positive	Negative
Predicted value	Positive	TP	FP
	Negative	FN	TN

of the CNN+LSTM model was set to 128, the maximum number of iterations was set to 300, the Adam optimizer was used, and the learning rate was set to 0.01. The experimental data were the top 5 vulnerabilities in the NVD, whose IDs, categories, and labels are presented in Table 2.

For feature extraction, the cleaned corpus was divided into a training set, a validation set, and a test set in the ratio of 6:2:2, and $\vartheta = 0.0005$ was set to screen feature words. A 50-dimensional feature word vector was obtained using the skip-gram model. After exponential weighting, the final word vector was used as the feature during vulnerability detection. The vulnerability detection performance of the model was evaluated using the confusion matrix (Table 3).

(1) $Recall = TP/(TP + FN)$
(2) $Precision = TP/(TP + FP)$
(3) $F1 = (2 \times Precision \times Recall)/(Precision + Recall)$
(4) Mathews correlation coefficient (MCC) [27]: $MCC = (TP \times TN - FP \times FN)/\sqrt{(TP + FP)(TP + TN)(TN + FP)(TN + FN)}$

The F1 value is a comprehensive evaluation of the recall rate and precision, and the MCC is an evaluation of the balance of the model classification. For vulnerability detection, higher F1 and MCC values of the model indicate that the model was better at vulnerability detection.

4.2 Analysis of Results

The CNN+LSTM model was compared with other methods:

(1) Support vector machine (SVM) [28],

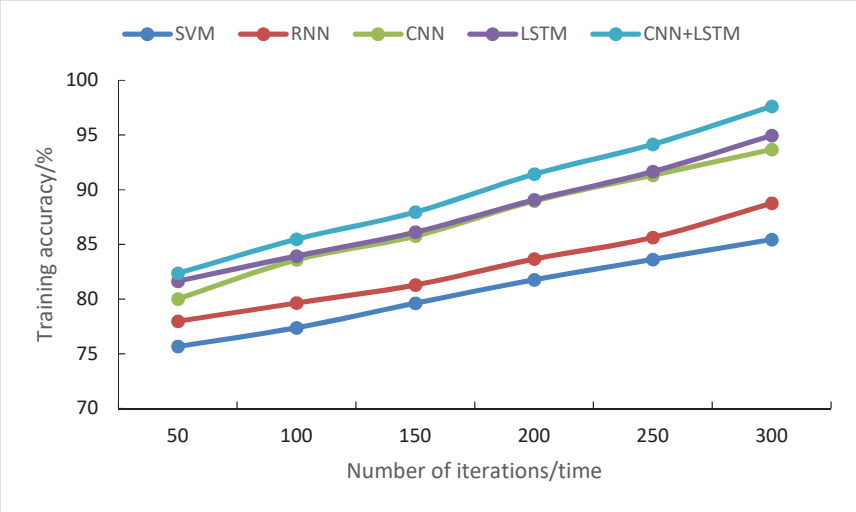

Figure 3 Comparison of the training accuracy between different methods.

(2) Recurrent Neural Network (RNN) [29],
(3) CNN [30],
(4) LSTM [31].

First, the comparison of the training accuracy between different methods on the training set is shown in Figure 3.

From Figure 3, it was seen that with the increase of iterations, the training accuracy of different methods was gradually improved, and the accuracy reached the highest when the number of iterations reached 300. After comparing different methods, it was found that the SVM algorithm had the lowest accuracy in detecting vulnerabilities (85.45% the highest), and the RNN algorithm also had a low accuracy in detecting vulnerabilities, below 90%. The difference between the training accuracy of the CNN and LSTM algorithms was small. When the number of iterations was 300, the training accuracy of the CNN and LSTM algorithms was 93.68% and 94.96%, respectively, while the accuracy of the CNN+LSTM method reached 97.63%, which was significantly higher than that of a single CNN or LSTM algorithm, proving the effectiveness of the hybrid model in vulnerability detection.

The performance of several different methods was compared on the test set, and the results are shown in Table 4.

From Table 4, on the test set, the vulnerability detection performance of the SVM algorithm was poor, its recall and precision were below 0.7, its F1

Table 4 Performance comparison of different methods on the test set

	SVM	RNN	CNN	LSTM	CNN+LSTM
Recall rate	0.6784	0.7012	0.7536	0.7607	0.8876
Precision	0.6892	0.7126	0.7654	0.7738	0.8739
F1 value	0.6838	0.7069	0.7595	0.7672	0.8807
MCC	0.9156	0.9367	0.9678	0.9687	0.9738

Table 5 Comparison of F1 values of different methods in detecting different categories of vulnerabilities

Vulnerability ID	SVM	RNN	CNN	LSTM	CNN+LSTM
CWE-79	0.6736	0.7001	0.7768	0.7762	0.8907
CWE-119	0.6864	0.7154	0.7825	0.7684	0.8867
CWE-20	0.7012	0.7126	0.7263	0.7536	0.8536
CWE-200	0.6525	0.7005	0.7436	0.7864	0.8687
CWE-787	0.7053	0.7059	0.7683	0.7514	0.9038

value was only 0.6838, and its MCC value was 0.9156; the RNN algorithm was slightly better than the SVM algorithm in vulnerability detection, and its F1 value reached 0.7069, which was 0.0231 larger than the SVM algorithm. The performance difference between the CNN and LSTM algorithms in vulnerability detection was not significant, with F1 values of 0.7595 and 0.7672, and MCC values of 0.9678 and 0.9687, respectively. The recall rate of the CNN+LSTM hybrid model reached 0.8876, showing an improvement of 0.134 and 0.1269 over the CNN and LSTM algorithms, respectively, and its precision reached 0.8739, showing an improvement of 0.1085 and 0.1085 over the CNN and LSTM algorithms, respectively. In terms of F1 value, only the CNN+LSTM algorithm exceeded 0.8 among the compared methods, reaching 0.8807, and its MCC value was also the highest, reaching 0.9738, indicating that the CNN+LSTM algorithm had better performance in vulnerability detection and could detect different vulnerabilities more accurately with better classification balance.

The results of comparing the F1 values of different methods on different categories of vulnerability detection are shown in Table 5.

From Table 5, it was seen that the F1 values of the SVM algorithm ranged from 0.65 to 0.7 when detecting different vulnerability categories, among which the best detection effect was achieved for CWE-787, with an F1 value of 0.7053. The F1 values of the RNN algorithm were all around 0.7, among which the best detection performance was achieved for CWE-119, with an F1 value of 0.7154. The CNN and LSTM algorithms had the best detection

effect on CWE-119 and CWE-200, respectively, with F1 values of 0.7825 and 0.7864, respectively. Finally, the F1 values of the CNN+LSTM algorithm were all above 0.85, among which the best detection effect was achieved on CWE-787, and the highest F1 value was 0.9038. In general, although different methods performed differently in detecting different categories of vulnerability, the CNN+LSTM algorithm always maintained a high F1 value, indicating that the method maintained a high level of detection for different categories of vulnerabilities and could achieve good detection for different categories of vulnerabilities. Therefore, the CNN+LSTM algorithm could provide services for practical network security.

5 Conclusion

This paper focused on the detection of network security vulnerabilities and designed a CNN+LSTM method to achieve the extraction and matching of vulnerability features. Through experiments on the NVD vulnerability library, it was found that the CNN+LSTM method had the highest training accuracy and the best performance on the test set compared with the SVM and RNN methods, with F1 values reaching 0.8807 and the MCC value reaching 0.9738, both better than the single CNN and LSTM methods, proving the reliability of this hybrid model for further applications in practical network security vulnerability detection. However, the CNN+LSTM method also has some limitations, for example, the vulnerability dataset used for experiments was small, the types of vulnerabilities are few, and the data imbalance situation was not investigated. Therefore, in future research, the applicability of this method on larger and more complex datasets needs to be analyzed, and the data imbalance situation needs to be improved.

References

[1] M. R. Neamah, J. Kh-Madhloom, O. A. Hassen, Z. Z. Abidin, O. A. Hassen, 'Fuzzy Logic Integrated Security Aware Algorithm for Vulnerability Avoidance in Network Environment', J. Adv. Res. Dyn. Control Syst., 10(10-Special Issue), pp. 785–794, 2018.

[2] L. Wang, R. Abbas, F. M. Almansour, G. S. Gaba, R. Alroobaea, M. Masud, 'An empirical study on vulnerability assessment and penetration detection for highly sensitive networks', J. Intell. Syst., 30(1), pp. 592–603, 2021.

[3] J. Li, J. Chen, M. Huang, M. Zhou, W. Xie, Z. Zeng, S. Chen, Z. Zhang, 'An Integration Testing Framework and Evaluation Metric for Vulnerability Mining Methods', China Commun., 15(2), pp. 190–208, 2018.
[4] A. Abdul Rasheed, 'Vulnerability detection towards protecting intrusion by Social Network Analysis approach,' 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, pp. 1219–1224, 2021.
[5] K. Chhillar, S. Shrivastava, 'Vulnerability Scanning and Management of University Computer Network,' 2021 10th International Conference on Internet of Everything, Microwave Engineering, Communication and Networks (IEMECON), Jaipur, India, pp. 01–06, 2021.
[6] H. Hanif, M. Nasir, M. Ab Razak, A. Firdaus, N. B. ANuar, 'The rise of software vulnerability: Taxonomy of software vulnerabilities detection and machine learning approaches', J. Netw. Comput. Appl., 179(9), pp. 1–24, 2021.
[7] G. Lin, S. Wen, Q. L. Han, J. Zhang, Y. Xiang, 'Software Vulnerability Detection Using Deep Neural Networks: A Survey', P. IEEE, 108(10), pp. 1825–1848, 2020.
[8] Y. Xiao, Z. Xu, W. Zhang, C. Yu, L. Liu, W. Zou, Z. Yuan, Y. Liu, A. Piao, W. Huo, 'VIVA: Binary Level Vulnerability Identification via Partial Signature', 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 213–224, Honolulu, HI, USA, 2021.
[9] V. A. Minaev, I. D. Korolev, A. V. Mazin, S. A. Konovalenko, 'Model of vulnerability identification in unstable network interactions with automated system', Radio Ind. (Russia), 28(2), pp. 48–57, 2018.
[10] P. Nancy, S. Muthurajkumar, S. Ganapathy, S. V. N. Santhosh Kumar, M. Selvi, K. Arputharaj, 'Intrusion detection using dynamic feature selection and fuzzy temporal decision tree classification for wireless sensor networks', IET Commun., 14(5), pp. 888–895, 2020.
[11] J. Ren, Z. Zheng, Q. Liu, Z. Wei, H. Yan, 'A Buffer Overflow Prediction Approach Based on Software Metrics and Machine Learning', Secur. Commun. Netw., 2019, pp. 1–13, 2019.
[12] V. Kovtun, I. Izonin and M. Greguš, 'Model of Information System Communication in Aggressive Cyberspace: Reliability, Functional Safety, Economics', IEEE Access, 10, pp. 31494–31502, 2022.
[13] M. Alqarni, A. Azim, 'Software source code vulnerability detection using advanced deep convolutional neural network', CASCON '21:

Proceedings of the 31st Annual International Conference on Computer Science and Software Engineering, pp. 226–231, 2021.

[14] S. Chakraborty, R. Krishna, Y. Ding, B. Ray, 'Deep Learning Based Vulnerability Detection: Are We There Yet?', IEEE T. Software Eng., 48(9), pp. 3280–3296, 2022.

[15] K. Munonye, M. Peter, 'Machine learning approach to vulnerability detection in OAuth 2.0 authentication and authorization flow', Int. J. Inf. Secur., 21(2), pp. 223–237, 2022.

[16] M. Begum, M. Arock, 'Efficient Detection Of SQL Injection Attack(SQLIA) Using Pattern-based Neural Network Model', 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, pp. 343–347, 2021.

[17] H. C. Chen, A. Nshimiyimana, C. Damarjati, P. H. Chang, 'Detection and Prevention of Cross-site Scripting Attack with Combined Approaches', 2021 International Conference on Electronics, Information, and Communication (ICEIC), pp. 1–4, Jeju, Korea, 2021.

[18] B. Biswas, A. Mukhopadhyay, 'G-RAM Framework for Software Risk Assessment and Mitigation Strategies in Organizations', J. Enterp. Inf. Manag., 31(2), pp. 276–299, 2018.

[19] N. Bi, J. Tan, J. H. Lai, C. Suen, 'High-Dimensional Supervised Feature Selection via Optimized Kernel Mutual Information', Expert Syst. Appl., 108(OCT.), pp. 81–95, 2018.

[20] H. Vranken, H. Alizadeh, 'Detection of DGA-Generated Domain Names with TF-IDF', Electronics, 11(3), pp. 1–28, 2022.

[21] L. Nguyen, H. H. Chung, K. V. Tuliao, T. M. Y. Lin, 'Using XGBoost and Skip-Gram Model to Predict Online Review Popularity', SAGE Open, 10(4), pp. 215824402098331, 2020.

[22] J. Zhao, 'Efficiency of Corporate Debt Financing Based on Machine Learning and Convolutional Neural Network', Microprocess. Microsy., 83(no.1), pp. 1–5, 2021.

[23] V. Kumar, R. S. Singh, Y. Dua Y, 'Morphologically dilated convolutional neural network for hyperspectral image classification', Signal Process. Image, 101, pp. 116549–, 2022.

[24] S. O. Michael, S. S. Juan, E. Mit, 'Preliminary Evaluation of Convolutional Neural Network Acoustic Model for Iban Language Using NVIDIA NeMo', J. Telecomm. Inform. Technol., pp. 43–53, 2022.

[25] I. Shahin, N. Hindawi, A. B. Nassif, A. Alhudhaif, K. Polat, 'Novel dual-channel long short-term memory compressed capsule networks for emotion recognition', Expert Syst. Appl., vol. 188, pp. 1–52, 2022.

[26] K. S. Kumar, B. R. Gomathi, 'Forecasting Photovoltaic Power Output Using Long Short-Term Memory and Neural Network Models', Int. J. Eng. Appl., 10(2), pp. 116–125, 2022.
[27] K. Abhishek, G. Hamarneh, 'Matthews Correlation Coefficient Loss For Deep Convolutional Networks: Application To Skin Lesion Segmentation', 2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI), pp. 225–229, Nice, France, 2021.
[28] L. B. Zhang, F. Peng, L. Qin, M. Long, 'Face spoofing detection based on color texture Markov feature and support vector machine recursive feature elimination', J. Vis. Commun. Image R., 51(feb.), pp. 56–69, 2018.
[29] D. Alis, C. Alis, M. Yergin, C. Topel, O. Asmakutlu, O. Bagcilar, Y. D. Senli, A. Ustundag, V. Salt, S. N. Dogan, M. Velioglu, H. H. Selcuk, B. Kara, C. Ozer, I. Oksuz, O. Kizilkilic, E. Karaarslan, 'A joint convolutional-recurrent neural network with an attention mechanism for detecting intracranial hemorrhage on noncontrast head CT', Sci. Rep., 12(1), pp. 1–9, 2022.
[30] A. Sehgal, N. Kehtarnavaz, 'A Convolutional Neural Network Smartphone App for Real-Time Voice Activity Detection', IEEE Access, 6, pp. 9017–9026, 2018.
[31] P. Ramaraj, 'A Neural Network in Convolution with Constant Error Carousel Based Long Short Term Memory for Better Face Recognition', Turk. J. Comput. Math. Educ., 12(2), pp. 2042–2052, 2021.

Biography

Ying Xue, born in 1979, has received the master's degree from xi'an jiaotong University in 2007. She is working in Shaanxi Police College now as a associate professor. She is interested in network security.