Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 16, No. 2, 2025

Intelligent detection of network security vulnerabilities based on machine learning and penetration behavior analysis

Xiaoyang Qiao [1], Xin Ai [1], Yiping Wu [1], Shuo Zhang [1], Bei Hu [2], Yunxia Yu[3, *]

[1] School of Artificial Intelligence, Jingchu University of Technology, Jingmen, Hubei, 448000, China
[2] School of Mathematics and Physics, Jingchu University of Technology, Jingmen, Hubei, 448000, China
[3] Intelligent Information Technology Research Center, Jingchu University of Technology, Jingmen 448000, China

[*] **Corresponding author:** 837133720@qq.com

Abstract: With the increasing frequency of network attacks, the traditional methods of network vulnerability detection are faced with many challenges. Within this manuscript, a network vulnerability detection framework, utilizing a deep neural network (DNN) that amalgamates advanced deep learning techniques with conventional machine learning methodologies, is posited to enhance discernment capabilities when pinpointing potential threats within network data streams. Through the nonlinear combination of multi-layer neurons, the model can effectively capture complex feature relationships, so as to achieve accurate classification of normal traffic and attack traffic. The model's structure encompasses an input stratum, manifold concealed strata, and an outer stratum. The ReLU activation function is harnessed to amplify the model's nonlinearity, incorporating the Sigmoid activation function in the latest layer augments the process of binary categorization. The choice of binary cross entropy as the loss function is made to refine the model's performance. The experimental blueprint ensures data integrity and dependability via the procedural measures of data prepossessing, feature culling, and normalization. In the training process, the weight of the model is constantly updated by backpropagation algorithm to improve its generalization ability on unknown data.

Keywords: Machine Learning; Network Security; Penetration Testing.

1. Introduction

Amidst the swift evolution of information technology, the issue of cybersecurity has been increasingly brought to the fore. With the popularization of the Internet and the acceleration of digital transformation, the frequency and complexity of various cyber attack methods have increased significantly, bringing severe challenges to the security of individuals, enterprises and even countries [1-2]. Malicious actors leverage sundry loopholes to assail systems, thereby precipitating a cascade of grave ramifications, including data exfiltration, pecuniary depletion, and reputational detrition. Traditional methods of network vulnerability detection, such as rule-based detection and signature matching, have played an important role in the early stage, but with the continuous progress of attack technology, these methods have gradually revealed their limitations. These methods usually rely on predefined rules and features, and have insufficient ability to identify new attacks or variant attacks, so the detection rate and response speed can not meet the actual needs in the face of complex network environments. In this case, there is an urgent need for a more advanced and efficient detection technology to meet modern network security challenges. In recent years, the burgeoning advancement of deep learning technology has infused novel prospects into the realm of network security. As a potent instrument within the machine learning paradigm, DNNS have garnered extensive application across diverse domains such as image recognition and natural language processing, owing to their multi-tiered architecture and formidable capacity for feature extraction. [3]. DNNS can automatically extract features by learning complex patterns in large amounts of data, enabling efficient classification and regression tasks. In network vulnerability detection, the application potential of DNNS is increasingly recognized by researchers because of their ability to process high-dimensional data and identify potential attacks from it. This study endeavors to proffer a cyber-vulnerability detection framework predicated upon deep neural networks, thereby augmenting the efficacy of discerning latent threats within network traffic. Through the nonlinear combination of multi-layer neurons, the model can effectively capture complex feature relationships, so as to achieve accurate classification of normal traffic and attack traffic. Specifically, the architecture of the model includes input layer, multiple hidden layers and output layer. The ReLU activation function is employed to bolster the model's capacity for nonlinear representation, while the Sigmoid activation function is utilized to execute dichotomous classification tasks within the output layer. The binary cross-entropy loss function is chosen to refine the model's performance. To substantiate the efficacy of the proposed model, the NSL-KDD dataset is selected as the experimental foundation. This data set is a classic data set within the purview of cyber-intrusion detection, including a variety of network attack types and normal traffic samples. Compared with its predecessor KDD Cup 1999 dataset, the NSL-KDD dataset solves the problem of data redundancy and imbalance, and provides a more realistic and diversified network traffic sample. Therefore, through in-depth analysis of this data set, the performance of the model in practical applications can be better evaluated. In order to ensure the validity and reliability of the data, a variety of data preprocessing measures are adopted in the experimental design, including feature selection and normalization steps. These steps not only improve the training efficiency of the

model, but also enhance the adaptability of the model to different types of data. In the course of training, the weight of the model is constantly updated by backpropagation algorithm to improve its generalization ability on unknown data. As the core algorithm in deep learning, backpropagation algorithm can optimize the model by calculating the gradient of the loss function and adjusting the model parameters according to the gradient. Through this iterative optimization process, the model can gradually learn the key features in the data and improve its identification accuracy of network attacks.

2. Related work

2.1. Vulnerability detection based on machine learning

Amidst the swift evolution of information technology, the issue of cyber security has ascended in prominence, most of the early studies focused on a single type of vulnerability detection, and relatively few comparative studies on different detection methods for the same data set. To fill this gap, a method combining machine learning and text mining has been proposed to predict components that may have security vulnerabilities in software applications [4]. The method analyzes the source code through text mining technology, and uses a bag of words model to treat software components as a series of terms with related frequencies and features them as predictors. The experimental results show that the combination of text mining and machine learning has a significant effect on improving the efficiency of detection of susceptibilities, and can significantly diminish the temporal and laborious demands for safeguarding software. In addition, there are studies that provide a high-quality public data set containing 223 vulnerabilities found in three Web applications [5]. Based on this data set, the researchers compared the vulnerability prediction model derived from the analysis of textual data with that based on software indicators, and the results showed that the text mining model was superior to the model based on software indicators in terms of recall rate.

2.2. Vulnerability detection technology of deep learning

In terms of vulnerability detection technology of deep learning, deep learning, as a popular multi-layer neural network machine learning method, has attracted much attention due to its efficient representation learning ability [6-9]. It can automatically find useful features in high-dimensional raw input without manual feature extraction, so it is very suitable for application in the field of vulnerability detection. In terms of feature extraction, common convolutional neural networks (CNNS) mainly consist of input layer, convolutional layer and pooling layer [10]. The input layer is responsible for entering data into the model; The convolutional layer extracts features through convolutional kernel, and selects effective information from complex input information. Pooling layer fuses the extracted features to extract significant features and achieve dimensionality reduction. The convolution layer can be divided into 1D-CNN, which is mainly used for processing one-dimensional data such as text, and only carries out convolution in the width direction. 2D-CNN is mainly used to process two-dimensional data such as images, and convolve width and height at the same time [8]. In addition, LSTM is also often used for vulnerability detection, which can capture long distance dependencies and is suitable for identifying logical features in code , shown in Figure 1[11]. Nevertheless, the comprehensive comprehension and empirical validation of the utilization of pertinent technologies in the realm of vulnerability identification remains lacking. As a result, the proposition has been put forward advocating for the amalgamation of convolutional neural networks (CNN), LSTM, and convolutional neural networks-long short-term memory networks (CNN-LSTM) in vulnerability detection [12].

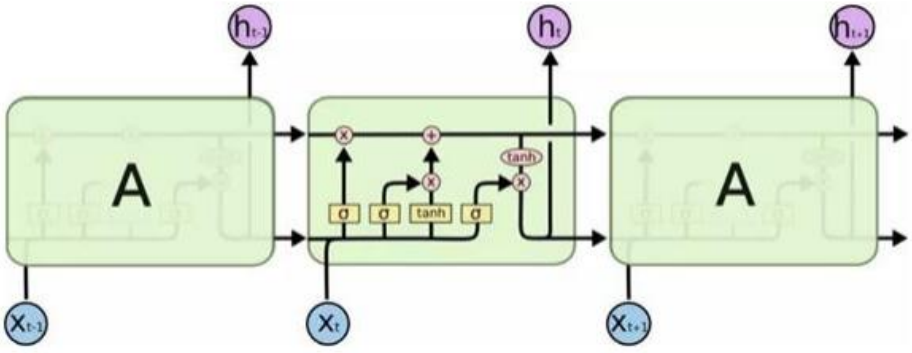

Figure. 1 Structure chart of LSTM

3. Model introduction

The following is a detailed introduction to the penetration testing section through the formula, combined with the design and training process of the deep neural network (DNN) model. Penetration Testing is a security assessment method that simulates attacks to identify and assess security vulnerabilities in a system. Through penetration testing, we can obtain information about the actual security status of the system, which can be input into the DNN model as a feature to improve the accuracy and reliability of vulnerability detection.

At the input level, we need to integrate a variety of characteristics, including source code characteristics, network traffic characteristics, and penetration test results. Assume that the input feature vector is:

$$\mathbf{x} = [\mathbf{x}_{code}, \mathbf{x}_{traffic}, \mathbf{p}] \tag{1}$$

$\mathbf{x}_{code}$ is a source code feature vector with dimension n_1 ,

$\mathbf{x}_{traffic}$ is a network traffic feature vector with dimension

n_2,

$\mathbf{p}$ is a penetration test feature vector with dimension m.

Therefore, the total dimension of the input vector is: $n = n_1 + n_2 + m$.

The model contains multiple hidden layers. The output calculation formula of layer l is as follows:

$$\mathbf{h}^{(l)} = f(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) \tag{2}$$

$\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times d_{L-1}}$ is the weight matrix of the l layer, where d_l is the number of neurons in the l layer and d_{l-1} is the number of neurons in the previous layer. $\mathbf{b}^{(l)} \in \mathbb{R}^{d_l}$ is the first layer of the bias vector l. f is the activation function (such as ReLU), defined as:

$$f(x) = \max(0, x) \tag{3}$$

After a hidden layer, add a penetration test module. Suppose the penetration test feature is introduced after layer l, combined with the penetration test feature $\mathbf{p}$:

$$\mathbf{h}^{(l+1)} = f(\mathbf{W}^{(l+1)}[\mathbf{h}^{(l)}, \mathbf{p}] + \mathbf{b}^{(l+1)}) \tag{4}$$

here, the combined input is:

$$[\mathbf{h}^{(l)}, \mathbf{p}] \in \mathbb{R}^{d_l + m} \tag{5}$$

$\mathbf{W}^{(l+1)} \in \mathbb{R}^{d_{l+1} \times (d_l + m)}$ is the first $l+1$ layer weight matrix.

$\mathbf{b}^{(l+1)} \in \mathbb{R}^{d_{l+1}}$ is the first $l+1$ layer of the bias vector.

The output layer is used to classify and output the probability distribution of vulnerabilities. The formula is:

$$\mathbf{y} = \sigma(\mathbf{W}^{(out)}\mathbf{h}^{(L)} + \mathbf{b}^{(out)}) \tag{6}$$

$\mathbf{W}^{(out)} \in \mathbb{R}^{k \times d_L}$ is the weight matrix of the output layer, k is the number of vulnerability categories, and d_L is the number of neurons in the last layer. σ is the softmax function, defined as:

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{k} e^{z_i}} \tag{7}$$

We use the cross-entropy loss function to optimize the model:

$$\mathcal{L} = -\sum_{i=1}^{m}\sum_{j=1}^{k} y_{ij} \log(\hat{y}_{ij}) \tag{8}$$

where y_{ij} is the true label (one-hot encoding) and $\hat{y}_{ij}$ is the probability predicted by the model. In the training process, the features extracted from the penetration test module are used to enhance the learning effect of the model. Calculating the gradient of the loss function and update the weights and bias using the backpropagation algorithm:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} \quad and \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} \tag{9}$$

the update rules are:

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} \mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} \tag{10}$$

here η is the learning rate.

The training process of the model is optimized using standard backpropagation algorithm combined with appropriate loss functions such as cross-entropy loss. The loss function can be expressed as:

$$L(\mathbf{y}, \mathbf{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \tag{11}$$

where N is the number of samples, y_i is the true label, and $\hat{y}_i$ is the probability predicted by the model. The structure diagram is shown in Figure 2.

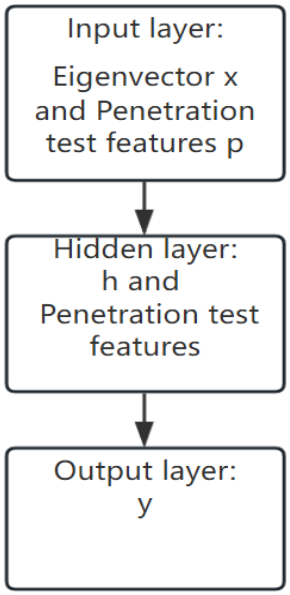

Figure. 2 Structure diagram of DNN model with penetration test

4. Experimental results and analysis

4.1. Model performance comparison

Within experiment A, a dataset was curated by tapping into the CVE database and enlisting feature data stemming from penetration testing, subsequent to which it was partitioned randomly into a training ensemble (70%), a validation ensemble (15%), and a testing ensemble (15%). In the course of model training, cross-validation is harnessed to uphold the capacity for generalization. The evaluation metrics, including Accuracy, Precision, Recall, and F1-score, will be used to evaluate the model's performance on different types of vulnerability detection tasks, specifically for both known and unknown vulnerabilities. In this experiment, we compare the performance of a vulnerability detection model based on deep neural networks (DNN) with our proposed improved model combined with a penetration test module (Ours). The empirical findings are detailed in Table 1:

Table 1. Model performance comparison

Experimental index	***Traditional DNN Model***	***Ours***
Accuracy (%)	91.52	92.51

Precision (%)	92.85	94.38
Recall (%)	88.64	89.02
F1-score (%)	89.87	90.45

In this experiment, we compare the performance of a traditional vulnerability detection model based on deep neural networks (DNN) with our proposed improved model combining penetration testing modules (Ours). The experimental results indicate that the enhanced model surpasses the conventional DNN model in multiple pivotal metrics. Specifically, in terms of accuracy, the DNN model was 91.52%, while our model improved to 92.51%, indicating that after the introduction of penetration testing, the model has increased its overall detection capability and can more effectively identify the right vulnerabilities and normal traffic. In terms of accuracy, the DNN model was 92.85%, while our model improved to 94.38%, which means that out of all samples detected as vulnerabilities, the proportion of true vulnerability samples increased, indicating that our model performed better in reducing false positives. In terms of recall rate, the DNN model is 88.64%, while our model is 89.02%, which is a smaller increase, but shows improvement in catching real vulnerabilities. Finally, in the comprehensive index of F1-score, the DNN model is 89.87%, while our model is improved to 90.45%, reflecting the superior performance in processing unbalanced data sets.

4.2. Model performance analysis under different network attacks

The aim of this experiment was to assess the efficacy of DNN models and penetration testing methodologies in detecting diverse forms of cyber assaults. The experimental setup consisted of training two models: a traditional deep neural network and a deep neural network enhanced by penetration testing. The data sets used for training include SQL injection, cross-site scripting (XSS), and samples of file inclusion attacks. Evaluation metrics include accuracy, robustness, utility, and response time to ensure a comprehensive evaluation of model performance. Parameters such as a learning rate of 0.001, batch size of 32 and 50 EPOchs are carefully configured to optimize training efficiency. The accuracy of each model will be compared to determine the percentage of improvement achieved through integrated penetration testing. In addition, response times will be measured to assess the practical applicability of the enhanced model in real-time scenarios. The empirical findings are detailed in Table 2.

Table 2. Performance of the model under different network attacks

Attack Type	Traditional DNN Model	Ours
SQL Injection (%)	78.5	82.3
Cross-Site Scripting (XSS) (%)	82.0	91.0
File Inclusion (%)	75.0	78.5
Overall Accuracy (%)	80.0	85.0
Response Time(ms)	210	160

It can be seen from the experimental results that the deep neural network (DNN) model combined with penetration testing shows significant advantages in various attack detection. Against SQL injection attacks, the detection accuracy of the traditional DNN model is 78.5%, while our model is improved to 82.3%, showing a significant improvement. This improvement not only reflects the model's better ability to recognize complex attack patterns, but also indicates that the introduction of penetration testing effectively enhances the model's learning ability. For cross-site scripting attacks (XSS), the accuracy of the traditional model is 82.0%, while our model reaches 91.0%, which also reflects the important role of penetration testing in feature extraction and attack identification. In terms of file inclusion attacks, the detection rate of the traditional model was 75.0%, while our model improved to 78.5%, further demonstrating the effectiveness of the new method. Overall, the overall accuracy was improved from 80.0% of traditional DNNS to 85.0%, showing a significant performance improvement. In addition, in terms of response time, the response time of the traditional model is 210 ms, while our model is optimized to 160 ms, indicating that the utility and response speed of the model are also improved while the detection capability is improved. These results show that the combination of penetration testing and DNN not only improves the detection capability of various attack types, but also keeps the latency low in practical applications, providing stronger support for network security protection.

4.3. Effect of feature selection method on model performance

This investigation endeavors to explore the influence of diverse feature selection techniques on the efficacy of an enhanced Deep Neural Network (DNN) architecture in the realm of network intrusion detection. The empirical data is derived from the publicly accessible CICIDS 2017 network traffic dataset, which encompasses a myriad of network assault types alongside benign traffic samples, rendering it apt for vulnerability identification tasks. To enable proficient model training and assessment, the dataset is initially partitioned into training, validation, and testing subsets, with the training subset constituting 70%, the validation subset 15%, and the testing subset 15%. This stratification guarantees the model's ability to generalize on previously unseen data. The model undergoes training utilizing a refined DNN framework, applying cross-entropy as the loss function and employing the Adam optimizer for parameter updates. Throughout the training phase, the validation subset is utilized to oversee the model's performance, thereby mitigating the risk of overfitting. The performance metrics for evaluation encompass accuracy, precision, recall, and F1 score, which collectively provide a comprehensive assessment of the model's performance across various network attack scenarios. The findings from the experiments are delineated in Table 3.

In the experiment of feature selection on the improved deep neural network (DNN) model, the performance difference of different methods is obvious. Although the correlation coefficient method can provide basic accuracy, it is relatively weak in terms of recall and accuracy. In contrast, Chi-square test significantly improved the model's performance, especially in the recall rate and accuracy rate, showing its effectiveness in feature screening. The most prominent is the random forest feature importance method, which not only has the best performance in accuracy, recall and accuracy, but also shows high efficiency in training and response time. This shows that the method can effectively identify the features that have the greatest impact on the model performance, and improve the overall detection capability. Although the

training time of principal component analysis is the shortest, its performance is not better than other methods, indicating that the information in the data may not be fully utilized in the feature selection. To sum up, the random forest feature importance method is superior in this experiment.

Table 3. Effect of feature selection method on model performance

Method	Correlation Coefficient Method	Chi-Square Test	Random Forest Features Importance	Principal Component Analysis
Accuracy (%)	86.5	88.0	89.5	87.3
Recall (%)	83.2	86.5	87.8	84.9
Precision (%)	85.0	87.2	88.6	86.0
Training Time (s)	30	28	25	20
Response Time (ms)	155	150	145	160

5. Summary

Under the background of the increasingly severe network security situation, the means of network attack are constantly evolving, and the traditional vulnerability detection methods are faced with many challenges. Hence, it is imperative to cultivate a proficient and precise vulnerability detection framework. This research aims to explore a new way to improve the capability of network security vulnerability detection by combining penetration testing and DNN. In this paper, a deep neural network model combined with penetration testing is proposed to improve the accuracy and efficiency of network security vulnerability detection. Through the experimental analysis of the NSL-KDD dataset, the results show that the improved model is superior to the traditional DNN model in several key indicators, showing significant improvement in detection ability. In addition, in the experiments targeting different types of network attacks, the model combined with penetration testing showed stronger recognition ability in the detection of multiple attack types, indicating that the introduction of penetration testing effectively enhanced the learning effect of the model. Simultaneously, the response time of the improved model is optimized, which shows that the response speed and practicability of the model in practical application are also improved while the detection capability is improved. Ultimately, a comparative analysis of the impact of various feature selection techniques on the efficacy of the model is conducted.

Acknowledgments:

Innovation and entrepreneurship training program for college students in Jingchu University of Technology (S202411336013)

References

[1] Liu Z. Intelligent classification of computer vulnerabilities and network security management system: Combining memristor neural network and improved TCNN model. PloS one. 2025 Jan 27;20(1):e0318075.

[2] Liu Z. Intelligent classification of computer vulnerabilities and network security management system: Combining memristor neural network and improved TCNN model. PloS one. 2025 Jan 27;20(1):e0318075.

[3] Arreche O, Abdallah M. A Comparative Analysis of DNN-based White-Box Explainable AI Methods in Network Security. arxiv preprint arxiv:2501.07801.

[4] Wu C, Wu F. Design and Implementation of Node Degree Centrality Computing of Network Security Database Based on Knowledge Graph. Security and Privacy. 2025 Jan;8(1):e473.

[5] Olutimehin AT. The Synergistic Role of Machine Learning, Deep Learning, and Reinforcement Learning in Strengthening Cyber Security Measures for Crypto Currency Platforms. Deep Learning, and Reinforcement Learning in Strengthening Cyber Security Measures for Crypto Currency Platforms (February 11, 2025). 2025 Feb 11.

[6] Y. Lecun, Y. Bengio, G. Hinton. Deep learning[J]. Nature, 2015, 521(7553): 436-444.

[7] Wang S, Liu T, Tan L. Automatically learning semantic features for defect prediction[C]//Proceedings of the 38th International Conference on Software Engineering, 2016: 297-308.

[8] Y. Shi , Y. E. Sagduyu , K. Davaslioglu . Vulnerability Detection and Analysis in Adversarial Deep Learning[J]. Guide to Vulnerability Analysis for Computer Networks and Systems: An Artificial Intelligence Approach, 2018: 211-234.

[9] P. Tsankov, A. Dan , D. Drachsler-Cohen. Securify: Practical Security Analysis of Smart Contracts[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018: 67-82.

[10] Markkandeyan S, Ananth AD, Rajakumaran M, Gokila RG, Venkatesan R, Lakshmi B. Novel hybrid deep learning based cyber security threat detection model with optimization algorithm. Cyber Security and Applications. 2025 Dec 1;3:100075.

[11] Battistello A, Bertoni G, Corrias M, Nava L, Rusconi D, Zoia M, Pierazzi F, Lanzi A. Unveiling ECC Vulnerabilities: LSTM Networks for Operation Recognition in Side-Channel Attacks. arxiv preprint arxiv:2502.17330. 2025 Feb 24.

[12] Hussain H, Kainat M, Ali T. Leveraging AI and Machine Learning to Detect and Prevent Cyber Security Threats. Dialogue Social Science Review (DSSR). 2025 Jan 22;3(1):881-95.