Journal of Combinatorial Mathematics
and Combinatorial Computing
www.combinatorialpress.com/jcmcc

Combinatorial Press

Research on Digital Campus Network Security Vulnerability Identification System Based on Numerical Analysis

Haibo Ji[1,✉], Kai Wang[2]

[1] *Information Technology Center, Chaoyang Normal University, Chaoyang, Liaoning, 122000, China*
[2] *Music Department, Chaoyang Normal University, Chaoyang, Liaoning, 122000, China*

ABSTRACT
In the digital campus network security construction, the existence of potential security vulnerabilities can easily cause serious threats to campus information security, resulting in significant losses. In order to prevent and mitigate the risk, the article designs a security vulnerability identification system. Firstly, the URL similarity is compared by machine learning in order to scan the vulnerability information. The SeCF embedding layer is utilized to improve the input speed and the discard layer is designed to improve the overfitting problem during the training process. Finally, TextACBL security vulnerability identification model is proposed by combining CA, 1D-CNN and BiLSTM techniques and analyzed numerically. The average recognition rate of this paper's method is as high as 80% for 10 common security vulnerabilities, which achieves better security vulnerability recognition results compared with existing methods such as cppcheck, deepbugs, flawfinder and vuldeepecker. The experimental results verify the effectiveness and feasibility of the method in this paper, which provides ideas for safeguarding campus network security during the construction of digital campus.

Keywords: textACBL, SeCF, BiLSTM, vulnerability identification, campus network security

1. Introduction

At present, there are many problems in the design and management of informationized education platforms in colleges and universities, and network security problems are particularly prominent. The Internet is an indispensable element of people's lives today. In the network, people connect with each other across the obstacles of time and space, and there are certain risks behind the connection [1-2]. In the design and construction of digital university platforms, network security events such as telecommunication fraud, information leakage, and equipment poisoning have emerged, which seriously damage the personal privacy and property safety of teachers and students [3-6]. At the same time, network attacks, data theft, website penetration, phishing emails and other behaviors have also

✉ Corresponding author.
E-mail address: cyncjhb@163.com (H. Ji).
Received 30 September 2024; Revised 20 November 2024; Accepted 15 February 2025; Published Online 16 April 2025.
DOI: 10.61091/jcmcc127b-040

brought certain challenges to the healthy, safe and orderly development of colleges and universities, and impeded the deepening of reforms and strengthening of governance for digital colleges and universities [7-9]. Network security work is an important work of educational informatization in colleges and universities, and it is an important link for colleges and universities to do a good job of maintaining security and stability, and the construction of a digital college platform needs to improve the management system of network security work through a variety of means and build a campus network security barrier [10-14]. Colleges and universities, as a talent training base for national strategic development, are also the front line of national network security [15-16]. It needs to pay great attention to information network security, strengthen network security vulnerability governance, and enhance the level of school network security protection has become a top priority. However, due to the special nature of the construction of smart colleges and universities, the large scale and other reasons, resulting in the lagging construction of its information network system, the number of old vulnerabilities, the high vulnerability level, the difficulty of processing and other problems are prominent, and it is urgent to further improve the network security protection capacity of colleges and universities [17-20].

In the context of informationization, the process of construction of digital campuses in colleges and universities is accelerating, and the problems of network security are becoming more and more prominent, and it is necessary to create a safe and reliable digital campus platform for teachers and students in colleges and universities to strengthen the risk management of network security, and to effectively prevent network security accidents from occurring [21]. Zheng, R. et al. investigated the hidden risks of the campus network security of a certain college or university using the WebHunt tool by constructing the knowledge The vulnerability intelligence database of the knowledge graph structure relies on the knowledge graph information matching function to locate network security vulnerabilities, and at the same time integrates functions such as domain name reverse resolution, which is better adaptable to network security detection in colleges and universities [22]. Awang, N., et al. proposed a method for identifying the risk of information security in colleges and universities based on the data mining method, which can be more adaptive to network security detection in colleges and universities than in universities and universities, by collecting the data of the intrusion into the defense system of the campus network and inputting it into a machine learning system for data analysis, which can better identify various types of information security threats related to the database [23]. Naagas, M. A. et al. applied the threat-driven approach to the cybersecurity analysis of higher education institutions, utilizing STRIDE and DREAD to identify potential threats to the cybersecurity of colleges and universities, and progressively strengthened the protection strength of the college and university cyberdefense system according to the security needs, thus reducing major security vulnerabilities [24]. Flores Jr, C. P. et al. used the open-source tool OWASP Zed Attack Proxy (ZAP) to analyze cybersecurity vulnerabilities in colleges and universities, and remediated cybersecurity deficiencies in colleges and universities based on the scanning results and reports to ensure that the risk of information on the websites related to educational digital platforms is minimized [25]. Awad, M. et al. used a college and university website page by inserting code into specific fields for vulnerability testing, utilizing Acunetix Web Vulnerability Scanning Tool (AWVS) to locate the cybersecurity vulnerabilities, and assessing the risk-threat intensity of each vulnerability as well as its impact on the website's security [26]. Liu, C. W. et al. assessed the role of centralized information technology (IT) governance processes in fending off cybersecurity intrusions in higher education institutions, and the study found that the governance effectiveness of centralized IT decision-making is positively correlated with the number of heterogeneous IT infrastructures in universities, providing insights into cybersecurity governance in universities [27].

The security vulnerability identification method designed in this research is divided into three parts. The first part focuses on resource provisioning and data updating for campus network vulnerability monitoring computation through machine learning methods, and comparing URL similarity to initially

scan vulnerability information. The second part uses SeCF embedding layer to improve the input efficiency, designs the discard layer to reduce the training overfitting, recombines the trained SeCF vector features, and constructs a security vulnerability identification model based on the long and short-term memory network model by using ReLU as the activation function. The third part extracts the semantic relations of feature words based on TD-IDF and mutual information methods, and introduces Embedding layer and negative sampling to shorten the Word2Vec training speed. Finally, CA, 1D-CNN and BiLSTM are fused to complete the security vulnerability recognition model construction, and security vulnerability recognition experiments are set up at the same time.

2. Campus network security vulnerability scanning system design

2.1. Monitoring system architecture design

The network security vulnerability monitoring system designed in this paper uses a 3-tier architecture as the basis to build the overall architecture of the system.

2.1.1. Display layer. The display layer of the security vulnerability monitoring system is to visualize the monitoring results, monitoring tasks and monitoring progress in the form of intuitive charts. System users can interact with the monitoring system by operating the corresponding buttons on the display layer interface.

2.1.2. Business logic layer. The business logic layer mainly processes and analyzes data related to network operation to obtain vulnerability monitoring results. Machine learning theory is used to process and analyze the data, and the scanning module scans the network vulnerabilities to locate the vulnerability information. In addition, due to the current network size, in order to reduce the imbalance in the distribution of system computing resources due to overloading of some of the computing nodes, the node load threshold is set to control the system computing resource equalization, which is calculated by the formula:

$$Y = \frac{\sum_{i=1}^{4} \xi_i (\gamma_i^{'} - \gamma_i)}{\sum_{i=1}^{4} (\gamma_i^{'} - \gamma_i)} \tag{1}$$

where: $\gamma_i (i = 1, 2, 3, 4)$ is the four metrics of computing resource utilization, node memory utilization, node communication interface utilization and node data throughput of the system computing nodes, respectively. $\gamma_i^{'}$ is the set upper limit value of the computing node load indicator. ξ_i is the threshold coefficient corresponding to each load indicator. When calculating the vulnerability monitoring data of a node in the business logic layer, the node allocates the calculation node according to the sequence set by the task manager. If the load occupied by the task that has been processed by the computing node reaches or approaches the threshold value, the computing node is deleted from the allocation sequence of the task manager to avoid prolonging the waiting period of the task and wasting computing resources.

2.1.3. Service access layer. The service access layer is connected to the network through a communication module to realize data interaction. In the service access layer, the relevant personnel can delete and change the vulnerability information data table, so as to update the vulnerability information and reduce the redundant operation of subsequent monitoring. The communication module in the service access layer is realized by using Twisted asynchronous communication network engine, which can ensure the safe transmission of response interaction data in the vulnerability monitoring process. The monitoring results of the machine learning module are stored in the central

database of the service access layer after data abstraction to update the network security vulnerability data samples.

2.2. Network Security Vulnerability Scanning

The vulnerability rules are read from the network security vulnerability rule base, and the basic information such as network inter-network interconnection protocol (IP) address and communication protocol are obtained by splitting the response packets. The similarity of network Uniform Resource Locator (URL) is used as the matching parameter of the aggregated hierarchical clustering algorithm, and the document object model tree is built through the page labels of the network. After that, the node weights corresponding to the DOM numbers of different web pages are assigned to get the URL similarity, which is calculated as:

$$sim(D_1, D_2) = sim(D_1T_{11}, D_2T_{11}) \tag{2}$$

$$\eta = \begin{cases} 1, (D_1T_{nm}, D_2T_{nm}) \\ 0, \text{other} \end{cases} \tag{3}$$

where: D is the DOM tree of the web page. T_{mm} is the tree node with depth n and node number m. η is the URL similarity. Store all URLs with similarity greater than 0.5 in the same subset and randomly select URLs to calculate their similarity with the remaining URLs. The two URLs greater than 0.5 are categorized into the same subset, and the calculation is repeated if they are less than 0.5. Based on the matching results of the aggregated hierarchical clustering algorithm, whether the network contains vulnerability information is obtained. If there are URLs that are similar to the vulnerability rules. It is determined that there is a vulnerability. According to the vulnerability scanning information use machine learning adaptive neural network model to monitor network security vulnerabilities.

3. Design and implementation of network code vulnerability identification model

3.1. Design of long and short-term memory network models

The designed long and short term memory network model is shown in Fig. 1, the model can be divided into SeCF input layer, LSTM layer, fully connected layer, discard layer and activation layer.

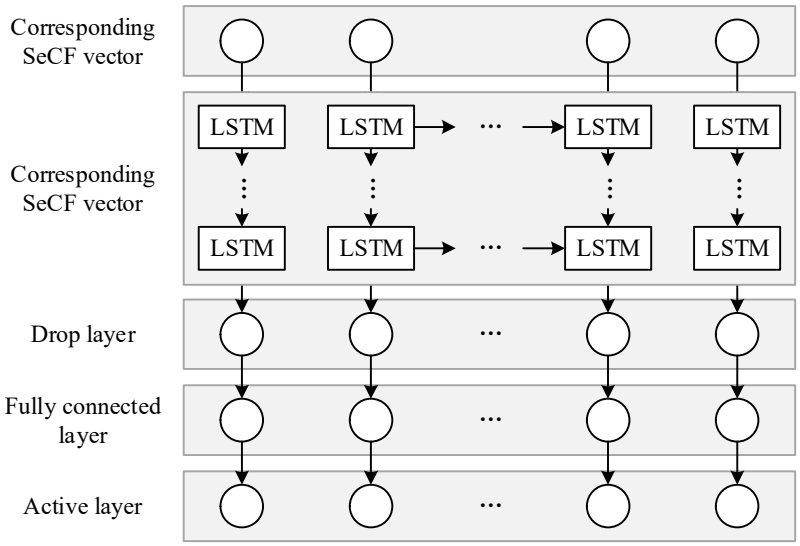

Fig. 1. LSTM network structure

3.1.1. SeCF Input Layer. The SeCF input layer is also called the embedding layer, and this layer first needs to convert the SeCF into vectors and downscale them. In traditional neural network models, such as Bag of Words (BOW) and One-Hot coding (One-Hot) methods are computationally inefficient. In this paper, the efficiency problem can be solved by using embedding layer. Embedding layer is actually a modification of bag-of-words model, which uses dense vectors to represent words and is able to automatically learn distributed representations from data. Each word is projected into a continuous vector space through its vector representation, and the position of the projection is obtained through text learning and will also be related to the words around that word. Embedding is the position of each word in the vector space [28].

Tensorflow provides an embedding layer for neural networks for text data. The input embedding layer data needs to be encoded using integers, so each word corresponds to an integer. This data preparation step can be performed using the embedding lookup provided by Tensorflow. The embedding layer will accomplish the task of embedding words in the dataset.

The embedding layer acts as the first layer of the model and it has to specify three parameters:

1) Input dimension: the size of the word in the text data.

2) Output dimension: the size of the vector space in which the words are embedded. The output dimension specifies the length of the output vector of the embedding layer of the word.

3) Input length: the length of the input sequence. In general, the embedding layer has learning weights and each embedded vector is updated during the training of the neural network.

3.1.2. LSTM layer. LSTM is a kind of deformed structure of RNN.LSTM units are generally based on ordinary RNN, and the memory unit is added to each neuron unit in the hidden layer, so as to control the memory information on the time series. Through the forgetting gate, input gate, candidate gate and output gate to control the degree of forgetting to pass information memory among the units in the hidden layer, so that the RNN network has a long-term memory function, for the practical application of RNN, has a huge role.

In the long and short term memory network model of this paper, the SeCF input layer data is written into the LSTM layer which passes through the output dimension of 16. In this layer, the degree of memorization and forgetting of information at each stage is controlled using LSTM unit controllable gates.

3.1.3. Discard layer. In general, in order to train a model, it is usually necessary to divide the dataset into a training set, a validation set and a test set. If there is an underfitting phenomenon it means that the validation set is poorly effective, while an overfitting phenomenon means that the validation set has good results but the test set is poorly effective. The main methods to reduce overfitting are Earlystopping, dataset augmentation, and model regularization (L1, L2). Discard Layer is also one of them, and its main idea is to train the deep neural network as an integrated model and then take half-means.

Discarding layers essentially regularize the parameters of the input and hidden layers. The regularized parameters make different subsets of the original neural network work better during training. For each input, there is a different neural network structure that shares the weights of the hidden nodes at the same time. Thus the samples and the model can then correspond to each other and overfitting can be avoided by the above method [29].

3.1.4. Fully connected layers. In the neural network model, the fully connected layer is mainly used as a classifier in the whole neural network model, i.e., the “distributed feature representation” obtained by the neural network model through learning is mapped to the sample labeling space. Generally speaking, in neural network models, each vector is individually feature extracted, i.e., local features are extracted. At this point, it is necessary to combine all the features into one piece. Therefore, in this paper, fully connected layers are utilized to accomplish this step.

In this paper, the fully connected layer recombines the features of each trained SeCF vector into one piece to form a new vector space. In other words, the SeCF vector is then processed by the data transformation and LSTM layer, and the neural network model reconverts the SeCF vector into a new vector based on the weight of each feature.

3.1.5. Activation layer. Each neuron in a neural network takes the output of the neuron in the upper layer as its input and passes it on to the next layer, along with the input attribute values. In multilayer neural networks, the functional relationship between the outputs of the nodes in the upper layer and the inputs of the nodes in the lower layer is realized by means of an activation function.

The ReLU used in this paper is a linear unsaturated activation function with the following formula:

$$tanh(x) = \frac{e^{x} + e^{-x}}{e^{x} - e^{-x}} \tag{4}$$

$$f(x) = \max(0, x) \tag{5}$$

1) ReLU solves the problem of vanishing gradients, and neurons do not saturate as long as x is in the positive interval.

2) Since ReLU is in linear non-saturated form, it can converge quickly in SGD.

3) It runs faster.ReLU has only linear relations and does not require exponential computation. The computation is faster than Sigmoid and Tanh [30], both in forward and backward propagation.

In a neural network model, the output of its activation layer is the output of the last time step. The closer this output is to 1, the more likely it is that the SeCF will be classified as containing a vulnerability.

3.2. Design and Implementation of Bidirectional Long and Short-term Memory Network Modeling

Bidirectional Long Short-Term Memory Networks not only have all the advantages of Long Short-Term Memory Networks, but also can take into account the contextual information of the vulnerable code, which is more suitable for detecting vulnerabilities in the code.

3.2.1. Design of bi-directional long and short-term memory network models. The bidirectional long and short-term memory network model is similar in structure to the long and short-term memory network model except that the LSTM layer is changed to BLSTM layer, mainly the neurons are different. In LSTM, the main memory is the historical input data and the current data, but most of the time when processing textual information also need to rely on the future input data, i.e., the contextual information are very critical. Therefore, scholars have proposed two-way neural networks. When performing neural network model training, the former LSTM neuron is responsible for processing forward data information, while the latter neuron is responsible for processing backward input information. The neuronal structure of BLSTM can be realized using the second neuron inversion [31].

3.2.2. Implementation of Bidirectional Long and Short-Term Memory Network Models. The forward LSTM is combined with the backward LSTM to form BiLSTM. Thus BiLSTM is implemented in a similar way to LSTM and will not be repeated here.The model diagram of BLSTM is shown in Fig. 2.

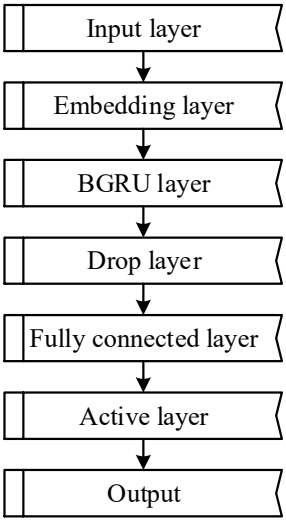

Fig. 2. BLSTM model

Some of the parameters for learning BLSTM are: batch size of 16, number of BLSTM cells of 128, output size of 256, training using minibatch stochastic gradient descent and ADAMAX, default learning rate of 0.002, maximum sequence length of 800, and discard rate of 0.2.

4. Digital campus network security vulnerability identification system design

The framework diagram of TextACBL vulnerability identification based on TFI-MI-W2V is shown in Fig. 3, and the whole vulnerability identification process consists of four phases: text preprocessing, text vector characterization, model training and vulnerability identification.

The vulnerability corpus needs to go through two stages of text preprocessing and text vector characterization before it is passed into the model for training, which converts the character-based text in the original vulnerability corpus into numerical vectors that can be trained by the model. The preprocessed vulnerability text takes into account the importance and relevance of the text feature words, and is characterized by the TFI-MI-W2V algorithm to realize the word-vector space conversion.

After obtaining the word vector space of different words through the TFI-MI-W2V algorithm, we unify the sentence length of the vulnerability text dataset to obtain the final target vulnerability text dataset, which is divided into training, validation and test sets according to the ratio of 6:2:2, where the training set is used to learn and train TextACBL vulnerability identification model, the validation set is used to find the optimal parameters of the model with the help of the cross-entropy loss function, and the test set is used to find the optimal parameters of the model based on different evaluation indexes. The training set is used to learn to train the TextACBL vulnerability recognition model, the validation set is used to find the optimal parameters of the model with the help of the cross-entropy loss function, and the testing set is used to evaluate the training effect of the TextACBL vulnerability recognition model based on different evaluation indicators.

The target vulnerability text dataset is converted into a word-vector two-dimensional matrix as the input to the TextACBL vulnerability recognition model and the data weights are shared. The TextACBL vulnerability recognition model acquires word vector spatial features, mines contextual semantic information, realizes automated vulnerability identification, and updates the model parameters with the help of the validation set by reducing the error between the predicted labels and the real labels obtained from the model training, so as to improve the vulnerability identification accuracy. The optimized model is used for testing and evaluation on the test set.

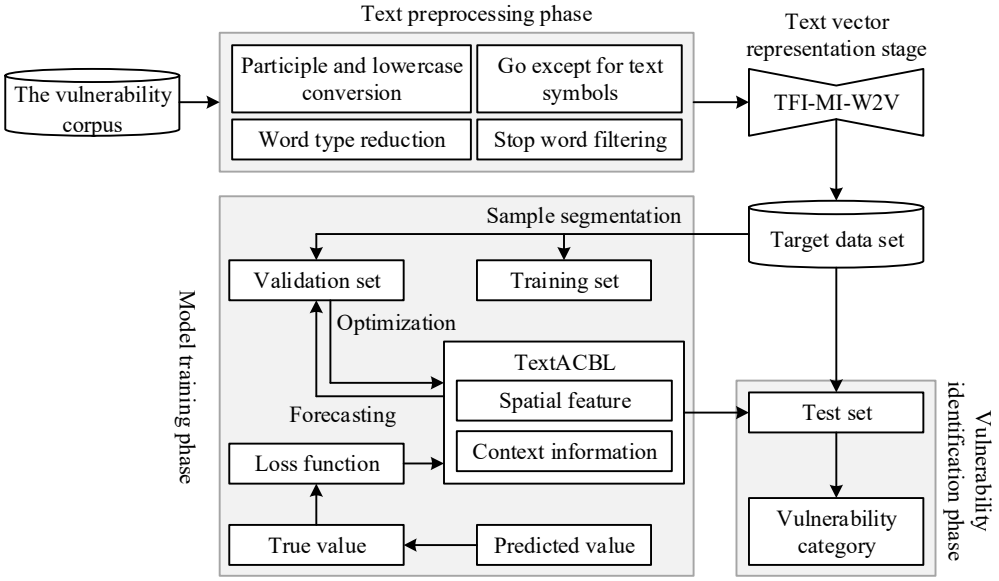

Fig. 3. TextACBL vulnerability identification framework based on TFI-MI-W2V

4.1. Text vector representation based on TFI-MI-W2V algorithm

4.1.1. TFI-MI keyword extraction. TFI-MI keyword extraction method is a hybrid keyword extraction algorithm based on TD-IDF and MI to calculate the weights of feature words, which aims to make up for the defects of TF-IDF such as ignoring the semantic relationship between words, and screen the key feature words affecting the text category, and the specific computational process is as follows:

1) Obtain three columns of text data information of text ID, text description and text category label from the corpus $D'_{n\times m}(m \geq 3)$ and perform text data preprocessing, and calculate the TF-IDF value $tf-idf(w_{i,i},d_i)$ of each feature word $w_{i,j}(1 \leq j \leq size(d_i))$ of each text $d_i(1 \leq i \leq n)$ in the corpus according to the cleaned corpus $D_{n\times 3}$.

2) The obtained TF-IDF values $tf-idf(w_{i,j},d_i)$ for each feature word are converted into a two-dimensional word space matrix $T_{n\times W}$ of n rows and W columns using solo thermal coding, where $W=\sum_{i}=1^{n} size(d_i), 1 \leq i \leq n, 1 \leq j \leq size(d_i)$. Each column represents the TF-IDF value of a feature word in n the text.

3) Merge the 2D matrix $T_{n\times W}$ with the text category labels and calculate the MI value $MI(w_{i,j},c)$ of mutual information of each feature word with the text category labels, where $w_i, i \in T$, T is the set of all feature words. Filter W feature words with mutual information $MI(w_{i,j},c)$ greater than the threshold value η affecting the text categorization keywords, where $\eta \in (0,1)$.

4.1.2. W2V Text Word Vector Representation. The cleaned corpus is used to mine the semantic relationship between words by measuring the sequence information between words through Word2Vec. The skip-gram model is selected by default in the code implementation process to realize the word embedding of the text corpus. In order to improve the network training speed and shorten the network training time, two improvements are realized for Word2Vec:

1) Introduce Embedding layer to reduce memory usage. When the vocabulary is increasing, the computational amount will also become very large. Therefore, we consider building an Embedding layer to replace the original MatMul layer to reduce the memory usage, and the Embedding layer only extracts the corresponding word row vector neurons from the weight matrix by specifying the index

number and passes them to the next layer, which greatly reduces the amount of computation and memory usage.

2) Introducing negative sampling to reduce computational effort. In order to reduce the computational amount, negative sampling is introduced to replace Softmax to keep the computational amount lower. Based on the word probability distribution N few negative examples n, $n \in N$ which are not related to the target word are randomly selected to reduce the sum of losses of positive examples and N negative examples to improve the accuracy of word vector, and the loss function is calculated as follows:

$$L = -\log \sigma(Y_T) - \sum_{n \in N} \log \sigma(Y_n) \tag{6}$$

where Y_T is the Embedding Dot layer output of positive examples and Y_n is the Embedding Dot layer output of negative examples. The negative sampling optimization can greatly reduce the computational complexity of the model and improve the training accuracy.

4.1.3. TFI-MI-W2V weighted calculation. The mutual information values $MI(w)$ of the keywords w learned from the TF-IDF 2D matrix are indexed and weighted with the word vectors $v(w)$ learned from Word2Vec to obtain the final keyword w word vectors $V(w)$, which are computed as follows:

$$V(w) = v(w) \cdot e^{MI(w)} \tag{7}$$

4.2. Vulnerability Identification Model Based on TextACBL

The TextACBL vulnerability identification model incorporates CA, 1D-CNN and BiLSTM, where CA and 1DCNN are used to obtain the spatial features of the vulnerability word vectors, and the extracted spatial features are passed into the BiLSTM module, which is used to mine the contextual semantic information of the vulnerability text, and finally outputs the vulnerability category of the text through Softmax activation function.

4.2.1. CA layer. Attention mechanism is an important tool used to improve the accuracy of feature extraction in vulnerability text recognition, in this paper, we propose a CA attention mechanism module so that it is embedded with a two-dimensional matrix input to obtain the key features of the vulnerability text word vectors that affect vulnerability classification.

Assuming that the CA inputs a 2D matrix of size $Y_{H \times W}$ (number of vulnerability text words H, size of vulnerability text word vector space W), 1D global average pooling and 1D global maximum pooling are performed on $Y_{H \times \pi}$, respectively, and the resulting feature matrices of the two $1 \times W$ are inputted into a two-tiered MLP where the parameter size of the first tier is 1/0. 125 = 8, and the parameter size of the second tier is 1. The two results are element-wise summed and the final CA1 feature matrix $A_{H \times W}$ is obtained by the sigmoid activation function, which is computed as follows:

$$\begin{aligned} F^{'} &= MLP(MaxPoollD(Y_{H \times W})) + MLP(AvgPoollD(Y_{H \times W})) \\ A_{H \times W} &= (a_{i,j}) = \sigma(F^{'}) \end{aligned} \tag{8}$$

where $MaxPoollD(\cdot)$ and $AvgPoollD(\cdot)$ represent one-dimensional global maximum pooling and one-dimensional global average pooling, respectively.

The channel note feature map $A_{H \times W}$ output from CA1 is vectorially multiplied with the input 2D matrix $Y_{H \times W}$ of the CA module to generate the feature matrix $Z_{H \times W} = A_{H \times W} \otimes Y_{H \times W}$ as the input to the CA2 module to obtain the 2D matrix longitudinal information. Global maximum pooling and global average pooling are done along the longitudinal axis for the input matrix $Z_{H \times W}$ and spliced, and the

final CA2 feature matrix $B_{H\times1}$ is obtained through a sigmoid activation function after a convolution operation with a filter size of 7 × 7, which is computed as follows:

$$\begin{aligned} F^{"} &= Conv^{7\times7}([MaxPool^{y}(Z_{H\times W}); AvgPool^{y}(Z_{H\times W})]) \\ B_{H\times1} &= (b_{i,j}) = \sigma(F^{"}) \end{aligned} \tag{9}$$

where $MaxPool^{y}(\cdot)$ and $AvgPool^{y}(\cdot)$ represent the global maximum pooling and global average pooling along the vertical axis, respectively. Finally, $Z_{H\times W}$ is vectorially multiplied with $B_{H\times1}$ to obtain the final output matrix $O_{H\times W}$.

4.2.2. Convolutional layers. The $O_{H\times W}$ 2D feature matrix extracted from the CA module is used as the input to the TextACBL vulnerability identification model, which contains only one convolutional layer, with a convolutional kernel size of 3×256, a step size of 1, and a filling mode of "same". Considering that the derivative value of the function is always 0 when $x<0$, which is easy to produce gradient dispersion, LeakyReLU is used as the activation function, and its function definition and derivative process are as follows:

$$\text{LeakyReLU}=\begin{cases} x & x\geq 0 \\ px & x<0 \end{cases} \tag{10}$$

$$\frac{d}{dx}\text{LeakyReLU}=\begin{cases} 1 & x\geq 0 \\ p & x<0 \end{cases} \tag{11}$$

where p is a hyperparameter with a value less than 1.

4.2.3. BiLSTM layer. Assuming x_t is the input sequence at moment t, where $t\in[1,N]$, then its hidden layer state h_t is defined as shown in Equation (12), where $\overrightarrow{\text{LSTM}}_t$ represents the forward hidden state of the BiLSTM at moment t, and $\overleftarrow{\text{LSTM}}_t$ represents the reverse hidden state of the BiLSTM at moment t, and the function definition of the two is shown in Equations (13)-(14), where $W_i(i\in[1,6])$ is the shared weights under different operations, and b is the bias. The t moment output y_t is shown in equation (15):

$$h_t = (\overrightarrow{\text{LSTM}}_t, \overleftarrow{\text{LSTM}}_t), t\in[1,N] \tag{12}$$

$$\overrightarrow{\text{LSTM}}_t = \tanh(W_1x_t + W_2\overrightarrow{\text{LSTM}}_{t-1} + b_{\overrightarrow{\text{LSTM}}}) \tag{13}$$

$$\overleftarrow{\text{LSTM}}_t = \tanh(W_3x_t + W_5\overleftarrow{\text{LSTM}}_{t+1} + b_{\overleftarrow{\text{LSTM}}}) \tag{14}$$

$$y_t = \tanh(W_4\overrightarrow{\text{LSTM}}_t + W_6\overleftarrow{\text{LSTM}}_t + b_y) \tag{15}$$

4.2.4. Layer Normalization Layer. The emergence of layer normalization layer LN effectively makes up for the defects of BN in RNN networks, and its normalization direction is perpendicular to BN each other, which can normalize the whole layer neurons of the current hidden layer without being affected by the size of the batch size, and its calculation process is as follows:

$$\hat{\mu}^{l} = \frac{1}{H}\sum_{i=1}^{H}x_i^{l} \tag{16}$$

$$\hat{\sigma}^{l} = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(x_i^{l}-\hat{\mu}^{l})^2} \tag{17}$$

$$\hat{x}^l = \frac{x^l - \hat{\mu}^l}{\sqrt{(\hat{\sigma}^l)^2 + \varepsilon}} \tag{18}$$

$$h^l = f(g^l \cdot \hat{x}^l + b^l) \tag{19}$$

Assuming that the input of layer l is x_i^l, the mean $\hat{\mu}^l$ and variance $\hat{\sigma}^l$ of layer l can be calculated by using the above equation, normalizing x_i^l, and finally the output of the LN is obtained by the activation function $f(\cdot)$ and the gain and bias parameters g^l and b^l.

5. Security vulnerability identification experiments

5.1. Experimental environment and experimental data

The network security vulnerability identification system proposed in this study integrates the use of multiple machine learning models and automatic machine learning optimization to improve the adaptive identification of vulnerabilities in different environments, and the implementation process of the model mainly consists of data processing, access, and implementation of algorithms.

5.1.1. Experimental environment. In order to carry out the subsequent experiments smoothly, this study built the hardware and software environments as shown in the following table, and the specific configurations are shown in Table 1.

Table 1. Experimental software and hardware information

Hardware	Parameter
Processor	Intel(R) Core(TM) i7-10700 CPU @
Machine band	RAM 32.0 GB
Hard disk	SSD 256GB + HDD 1TB
GPU	3060 mobile
Software	Function
Windows 11	64-bit operating system
Python	Development language
VScode	Development tool
TensorFlow	Depth learning framework
Keras	Depth learning framework
Sklearn	Machine learning framework

5.1.2. Introduction to the Security Vulnerability Dataset. The dataset used for training and testing the model in this research project is the IoT-23 dataset collected in a real IoT environment.

The IoT-23 dataset consists of data from 23 scenarios of different IoT network traffic. These scenarios are categorized into network traffic from 20 IoT devices containing vulnerabilities (pcap files, malware with corresponding names is executed in each scenario) and network traffic from 3 real IoT devices. In each scenario containing vulnerabilities, the experimenter executed a specific malware sample in the Raspberry Pi that used multiple protocols and performed different actions.

Table 2 shows a summary of the information for the twenty scenarios containing vulnerabilities. This includes the dataset name, duration, number of packets, number of packets in the log file, size of the original pcap file, and the possible name of the malware sample used to infect the device. Due to the long capture time of malicious exploits and the large amount of traffic generated by each exploit, the pcap file is rotated every 24 hours during data collection. However, in some cases, the pcap file grows too fast, which stops data collection before 24 hours. As a result, some data collection times will vary.

Table 2. The characteristics of the Internet of things networking

Name of Dataset	Duration(hrs)	Packets	ZeekFlows	Pcapsize	Name
32-1	24.3	222000	23154	132 MB	Mirai
45-1	2.5	85000000	6732218	8 GB	Mirai
46-1	3.8	1409000	286	1.8 GB	Mirai
41-1	8.6	19000000	5410654	1.1 GB	Mirai
53-1	25.0	62000000	19781482	4.9 GB	Mirai
22-1	24.8	60000	3300	3.4 MB	Torii
23-1	25.4	60000	3378	3.6 MB	Torii
44-1	9.5	21000	4427	2.1 MB	Trojan
63-1	24.2	288000000	3642	28 GB	Gagfyt
18-1	25.6	111000000	54659955	7.6 GB	Kenjiro
34-1	25.6	15000000	13645153	987 MB	Okiru
33-1	26.0	55000000	54454617	3.3 GB	Kenjiro
7-1	24.4	29000	10411	2.8 MB	Hakai
36-1	25.9	43000000	10504	3.7G	Mirai
47-1	24.4	15000000	3394369	1.3G	Mirai
35-1	7.1	76000000	73569013	5.8GB	IRCBot
9-1	24.9	12000000	11454782	878 MB	Linux,Mirai
8-1	25.8	6397000	6378298	478 MB	Linux.Hajime
3-1	36.7	484000	156133	51 MB	Muhstik
2-1	113.4	1671000	1008759	142 MB	Hide

5.1.3. Feature Extraction and Analysis. First of all, this experiment first loads all 23 datasets of the IoT23 dataset into data frames individually, skipping the first 10 rows and reading the last 200,000 rows when loading the data. Then all 23 data frames are combined into a new data frame. Next, variables that had no effect on the results were removed. These variables include: time of first packet (ts), connection identifier (uid), IP address, port number, application protocol, and connection status history. In addition, assign values to the transport layer protocol (proto) and connection status variables (cnn) and replace all missing values with 0. Finally, the combined dataset is generated and saved as a CSV file.

When performing real-time vulnerability identification, by using the import syntax, the processed anomaly data can be quickly deposited into the pre-defined vulnerability data table according to the CSV file format to realize the anomaly data dumping. When the training of the model or adaptive retraining of the model is required, the anomaly dataset can be exported to a CSV file by using the export syntax, which provides convenience to the training of the model and reduces the time of data processing.

For the selected CSV file data, the Sklearn package tool is used to analyze the network traffic data attribute distribution, vulnerability category distribution, and data item correlation in the IoT23 dataset.The distribution of the processed data attributes of the IoT23 dataset is shown in Figure 4.

The figure shows the distribution of values of different types of data items in the collected network traffic, in which the values of data items such as network protocol type, service protocol, connection time, packet size, return packet size, number of bytes lost, packet size correlation, etc., are all centrally distributed in a single fetch (or two fetches, but the other occupies a very small proportion). Such centrally distributed network attributes indicate that normal data and data containing vulnerabilities cannot be classified by analyzing these data items alone. On the other hand, attributes such as the port number of the destination address and the endpoint address of the connection have more types of numerical distributions, and the addition of such data items can be considered in the feature selection for subsequent training.

Fig. 4. Data property distribution

Figure 5 shows the distribution of vulnerability categories, demonstrating the percentage of the number of different types of network vulnerabilities in the IoT23 dataset. In terms of number normal traffic (Benign), DDoS vulnerabilities, and botnet vulnerabilities occupy a larger portion, while others such as different types of C&C vulnerabilities occupy a smaller portion. This data distribution is also a reflection of the distribution of vulnerabilities in the real world. In terms of vulnerability categories, the selected IoT23 dataset encompasses the mainstream network vulnerability types in the current

network environment including DoS, C&C, botnet, Trojan attacks, etc., which also reflects the applicability of this model in subsequent deployment applications in real-world environments.

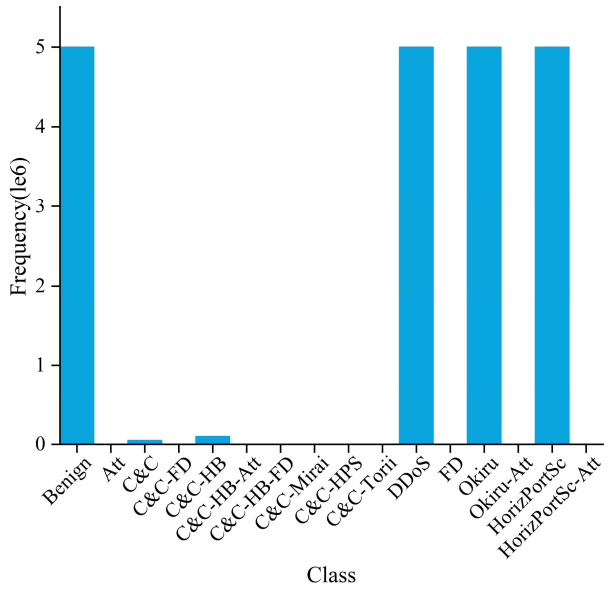

Fig. 5. Vulnerability category distribution

The correlation between different data items is shown in Fig. 6. The correlation between data items refers to the degree of association between a data item and other data items (calculated uniformly with the degree of connection between each data item and itself as 100%), and the closer the correlation implies the more similar the distribution of the data items. For example, the connection status data in the figure is not only closely related to itself (100%), but also closely related to historical connection status (58.52%) and connection IP (22.82%). When a data item has a higher degree of association with all other data items, the more the data item reflects the distribution characteristics of the overall dataset, and thus should be added to the feature set for subsequent model training.

In the model training of automatic machine learning tuning, this experiment chooses to input all the data attributes to be selected into the search space, and the optimal training data is automatically selected by automatic machine learning. In the model training with manual tuning, the 15 attribute data that are more relevant to vulnerability analysis in data analysis are artificially integrated into the training data set.

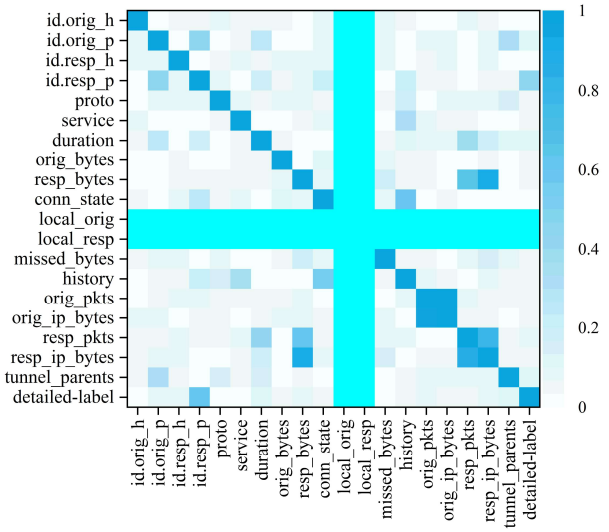

Fig. 6. Data correlation

5.2. Analysis of security vulnerability identification results

5.2.1. Evaluation indicators. In order to validate the experimental results and evaluate the effectiveness of the vulnerability identification method, this study uses four evaluation metrics to evaluate the results of the source code feature extraction method, which are precision, recall, F1 value and accuracy.

5.2.2. Analysis of experimental results. In order to verify the effectiveness of the vulnerability identification model proposed in this paper, this section uses the dataset in Section 5.1.2 to train and validate the vulnerability identification model. The dataset is divided into a training set and a validation set, and this paper chooses the ten-fold cross-validation method to ensure the accuracy of the experimental results, i.e., 9 out of every 10 copies of the data are used as the training set, and 1 copy is used as the validation set, with the ratio of the training set to the validation set being about 9:1. In order to further prove the effectiveness and superiority of this method over the traditional vulnerability identification methods, this paper also selects representative vulnerability identification tools at the present stage cppcheck, deepbugs, flawfinder and vuldeepecker model, the source code data set collected in this paper is input into the model to compare and analyze the difference between this model and the traditional vulnerability identification model.

In this paper, the dataset is first processed using sampling methods, i.e., undersampling and oversampling methods, and then the processed data is input into the model for training and testing, in order to reduce the impact of the imbalance of the dataset and to improve the model recognition effect.

To further validate the effectiveness of this vulnerability identification model, this paper trains and validates several vulnerability identification tools and models under the same criteria, namely cppcheck, deepbugs, flawfinder, and vuldeepecker, and the identification results obtained are shown in Figure 7.

The security vulnerability identification model proposed in this paper is significantly better than other mainstream vulnerability identification tools or methods. The recall of the vuldeepcker model is slightly higher than that of the present method because the precision is negatively correlated with the recall and the precision value obtained from the experiments of the present method is significantly higher than that of the vuldeepcker model, so a slightly lower recall may occur. vuldeepcker is a vulnerability identification model developed based on the semantic related code snippet codegadget, and its neural network model is implemented based on the Bi-LSTM bidirectional long short-term memory network. The method proposed in this paper uses attention network for feature learning while characterizing the source code feature information in a perfect way, which fully considers the structure and context information of the source code data, so the overall effect of the vulnerability identification task in this method is better than that of the vuldeepcker model.

deepbugs is a vulnerability identification method based on names such as variable names, function names, etc., which is based on semantic representations to reason about the names and identify vulnerabilities. deepbugs is overall based on natural language processing technology, which does not consider the contextual semantic information, and is unable to dig into the deeper syntactic structure of the source code, so all the indexes are lower than the present model.

flawfinder and cppcheck are traditional static vulnerability identification tools. Experiments show that the two identification tools have certain advantages in identification speed, but their identification effect is poor, and their precision and accuracy are below average. The reason for the poor recognition of flawfinder and cppcheck is that these vulnerability identification tools rely on fixed vulnerability patterns for vulnerability identification, and the vulnerability patterns rely on experts to manually analyze and construct them, so if there are vulnerabilities in the data to be identified that do not match the characteristics of the defined vulnerabilities, then the recognition results will be poor.

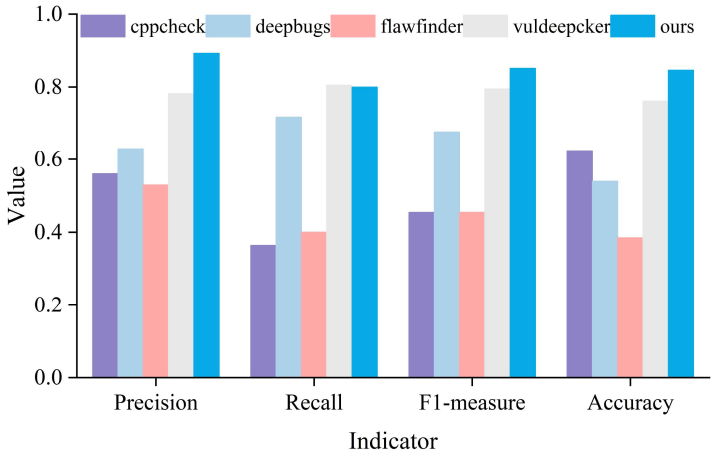

Fig. 7. Identification result

In this paper, the subject operating characteristic curve (ROC) is chosen to further validate the vulnerability identification effect of this method when the distribution of positive and negative samples in the dataset changes.The horizontal coordinate of the ROC curve is the false-positive rate FPR, and the vertical coordinate is the true-positive rate TPR.In the ROC curve graph, the more convergent to the upper-left corner of the curve corresponds to the model of the classification effect is more effective, and the higher the accuracy of vulnerability identification. The ROC curve graph obtained from the experimental analysis of this method with different vulnerability identification methods and tools is shown in Figure 8.

The ROC curve corresponding to the TFI-MI-W2V based vulnerability identification method is more convergent to the upper left corner compared to the other models, which provides better results in the vulnerability identification task. Compared with other vulnerability identification methods and tools, this method uses a more complete word vector characterization approach to characterize the source code and uses a BiLSTM-based learning module for classification, thus maintaining better vulnerability identification when the distribution of positive and negative samples changes. The ROC curves of vuldeepcker and flawfinder are closer and both perform poorly. Both vulnerability identification models rely on manually formulated vulnerability patterns, and thus are less effective in vulnerability identification when the positive and negative sample data are unbalanced. The performance of cppcheck and deepbugs is fair, but still lower than the vulnerability identification methods proposed in this paper, because the two vulnerability identification models only focus on the source code text information and do not take into account the code context semantic information, so their ROC curves are lower than those of the methods proposed in this paper.

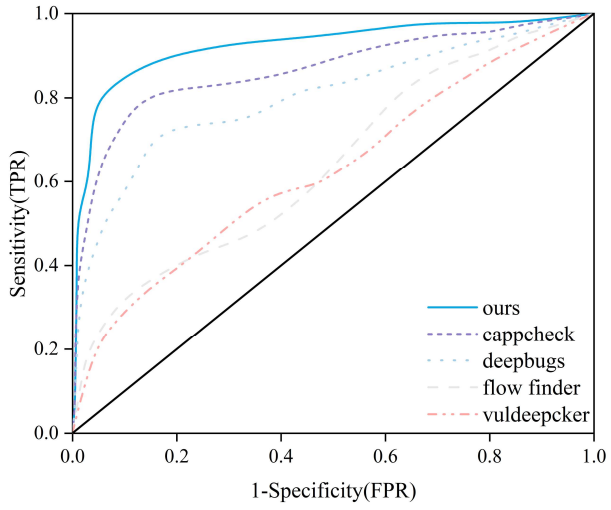

Fig. 8. ROC curve

In order to verify the recognition effect of this method for datasets of different vulnerability types, this paper uses source code datasets of different vulnerability types in the process of training and validation, and the results obtained from the experiments are shown in Figure 9.

This experiment extracted 10 common different vulnerability types of code composed of datasets for model training and testing, the experimental indicators are shown in Figure, in which the average value of precision and accuracy are higher than 80%, significantly better than the same type of vulnerability identification methods. This method can successfully identify vulnerability codes and classify the source code dataset during the experiment, and has good recognition effect for most of the common vulnerability type data.

False-positive samples refer to data that are incorrectly classified as positive samples, and the number of false-positive samples should be minimized. This model has high accuracy in the training and recognition process, while higher accuracy can reduce the false positive rate of the experiment, which can greatly shorten the time of manual verification and save resources such as manpower and material resources.C&C is a common type of buffer vulnerability, and traditional vulnerability recognition models have poor recognition effect on it. This method has a better effect on extracting the semantic information of the vulnerability and obtaining the context information of the vulnerability code, and it can extract the reference relationship between variables and other data, so as to identify the vulnerability such as the C&C type, and it has a better recognition effect.

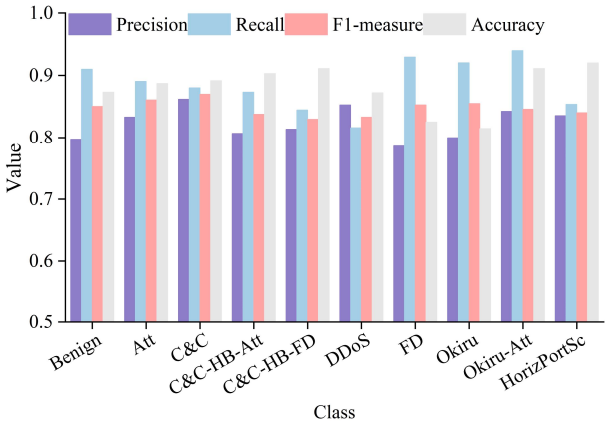

Fig. 9. Experimental result

According to the above experimental data, it can be seen that the vulnerability identification method based on TFI-MI-W2V proposed in this paper is effective in source code vulnerability identification and has certain superiority compared with traditional vulnerability identification methods, which can provide source code vulnerability identification results for researchers related to the network security of digital campuses.

6. Conclusion

In this paper, we integrate two-way long and short-term memory network, attention mechanism and one-dimensional convolutional neural network to detect vulnerabilities, propose a method for security vulnerability identification in digital campus, and numerically analyze the identification system to achieve better security vulnerability identification in IoT-23 dataset. The average recognition rate of 10 common security vulnerabilities is higher than 80%, and its overall performance is better than existing models such as cppcheck, deepbugs, flawfinder and vuldeepecker. The ROC curve of this method also achieves better vulnerability identification results when the distribution of positive and negative samples in the dataset varies. It shows that the security vulnerability

identification method designed in this study realizes the improvement of the existing methods, which is conducive to safeguarding campus information security.

Funding

This work was supported by 2022 Science Project for the 14th Five-year Plan of the Office for Education Science & Planning in Liaoning Province (NO: JG22EB020).

About the Authors

Haibo Ji was born in Chaoyang, Liaoning, P.R. China, in 1980. She obtained a master's degree from Dalian Jiaotong University in China. I am currently working at the School of Information Technology Center, Chaoyang Normal University.I am an associate professor. My main research direction is Network security and Construction of digital campus.

Kai Wang was born in Chaoyang, Liaoning, P.R. China, in 1978. He obtained a master's degree from Dalian Jiaotong University in China. I am currently working at the School of Music Department, Chaoyang Normal University. I am an associate professor. My main research direction is Education management and Ideological education.

References

[1] Singh, U. K., Joshi, C., & Gaud, N. (2016). Measurement of security dangers in university network. International Journal of Computer Applications, 155(1), 6-10.

[2] Adu-Boahene, C., Nikoi, S. N., & Nsiah-Konadu, A. (2021). Campus Network and Systems Security Assessment Using Penetration Testing: The Case of the University of Education Winneba, Kumasi. Asian Journal of Research in Computer Science, 12(1), 7-25.

[3] Li, J., Xiao, W., & Zhang, C. (2023). Data security crisis in universities: identification of key factors affecting data breach incidents. Humanities and Social Sciences Communications, 10(1), 1-18.

[4] Seneviratne, H. A., Thenabadu, M., & Wijerathne, W. M. G. K. (2022). An Exploratory Study on Information Security Vulnerabilities in Higher Education: Case of University of Vocational Technology, Sri Lanka. International Journal of Research and Innovation in Social Science, 6(6), 399-403.

[5] Liu, Y., Zhai, W., & Ji, S. (2021). Research on Campus Network Security Problem and Protection Strategy. In Research Anthology on Artificial Intelligence Applications in Security (pp. 1352-1369). IGI Global.

[6] Sembiring, J., Ramadhan, M., Gondokaryono, Y. S., & Arman, A. A. (2015). Network security risk analysis using improved MulVAL Bayesian attack graphs. International Journal on Electrical Engineering and Informatics, 7(4), 735.

[7] Abdulrasool, F. E., & Turnbull, S. J. (2020). Exploring security, risk, and compliance driven IT governance model for universities: applied research based on the COBIT framework. International Journal of Electronic Banking, 2(3), 237-265.

[8] Lamarca, B. I. (2020, February). Cybersecurity risk assessment of the university of northern Philippines using PRISM approach. In IOP Conference Series: Materials Science and Engineering (Vol. 769, No. 1, p. 012066). IOP Publishing.

[9] Sharma, A., Singh, U. K., Upreti, K., Kumar, N., & Singh, S. K. (2021, October). A Comparative analysis of security issues & vulnerabilities of leading Cloud Service Providers and in-house University Cloud platform for hosting E-Educational applications. In 2021 IEEE Mysore Sub Section International Conference (MysuruCon) (pp. 552-560). IEEE.

[10] Khader, M., Karam, M., & Fares, H. (2021). Cybersecurity awareness framework for academia. Information, 12(10), 417.

[11] Wang, J., Deng, G., Cai, R., Wu, Y., & Feng, H. (2023, June). Research on university network security situation assessment model. In International Conference on Computer Network Security and Software Engineering (CNSSE 2023) (Vol. 12714, pp. 15-21). SPIE.

[12] Bandara, I., Balakrishna, C., & Ioras, F. (2021). The need for cyber threat intelligence for distance learning providers and online learning systems. The Need For Cyber Threat Intelligence For Distance Learning Providers And Online Learning Systems, 9312-9321.

[13] Ramanauskaitė, S., Urbonaitė, N., Grigaliūnas, Š., Preidys, S., Trinkūnas, V., & Venčkauskas, A. (2021). Educational organization's security level estimation model. Applied Sciences, 11(17), 8061.

[14] Fang, J. (2022, December). Security evaluation method of distance education network nodes based on machine learning. In International Conference on Machine Learning for Cyber Security (pp. 281-295). Cham: Springer Nature Switzerland.

[15] Mangaoang, E. F., & Monreal, R. N. (2024). Common Vulnerabilities and Exposures Assessment of Private Higher Educational Institutions Using Web Application Security. Journal of Electrical Systems, 20(5s), 668-676.

[16] Shaorong, W., & Guiling, L. (2023). Research on campus network security protection system framework based on cloud data and intrusion detection algorithm. Soft Computing, 27(10), 6835-6844.

[17] Sánchez-Torres, B., Rodríguez-Rodríguez, J. A., Rico-Bautista, D. W., & Guerrero, C. D. (2018). Smart Campus: Trends in cybersecurity and future development. Revista Facultad de Ingeniería, 27(47), 104-112.

[18] DJEKI, E., DEGILA, J., BONDIOMBOUY, C., & ALHASSAN, M. H. (2021, November). Security issues in digital learning spaces. In 2021 IEEE International Conference on Computing (ICOCO) (pp. 71-77). IEEE.

[19] Cains, M. G., Flora, L., Taber, D., King, Z., & Henshel, D. S. (2022). Defining cyber security and cyber security risk within a multidisciplinary context using expert elicitation. Risk Analysis, 42(8), 1643-1669.

[20] Ibnugraha, P. D., Satria, A., Nagari, F. S., Rizal, M. F., & NonAlinsavath, K. N. (2023). The Reliability Analysis for Information Security Metrics in Academic Environment. JOIV: International Journal on Informatics Visualization, 7(1), 92-97.

[21] Singar, A. V., & Akhilesh, K. B. (2020). Role of cyber-security in higher education. Smart Technologies: Scope and Applications, 249-264.

[22] Zheng, R., Ma, H., Wang, Q., Fu, J., & Jiang, Z. (2021). Assessing the security of campus networks: the case of seven universities. Sensors, 21(1), 306.

[23] Awang, N., Samy, G. N., Hassan, N. H., Maarop, N., Magalingam, P., & Kamaruddin, N. (2020, May). Identification of information security threats using data mining approach in campus network. In Journal of Physics: Conference Series (Vol. 1551, No. 1, p. 012006). IOP Publishing.

[24] Naagas, M. A., & Palaoag, T. D. (2018, February). A threat-driven approach to modeling a campus network security. In Proceedings of the 6th International Conference on Communications and Broadband Networking (pp. 6-12).

[25] Flores Jr, C. P., & Monreal, R. N. (2024). Evaluation of Common Security Vulnerabilities of State Universities and Colleges Websites Based on OWASP. Journal of Electrical Systems, 20(5s), 1396-1404.

[26] Awad, M., & Ataelfadiel, M. (2024). Detecting vulnerabilities in universities' websites: A Security Analysis. Journal of Electrical Systems, 20(3), 2453-2461.

[27] Liu, C. W., Huang, P., & Lucas Jr, H. C. (2020). Centralized IT decision making and cybersecurity breaches: Evidence from US higher education institutions. Journal of Management Information Systems, 37(3), 758-787.

[28] Jaesin Ahn,Jiuk Hong,Jeongwoo Ju & Heechul Jung. (2023). Redesigning Embedding Layers for Queries, Keys, and Values in Cross-Covariance Image Transformers. Mathematics(8).

[29] Choe Junsuk,Lee Seungho & Shim Hyunjung. (2020). Attention-based Dropout Layer for Weakly Supervised Single Object Localization and Semantic Segmentation.. IEEE transactions on pattern analysis and machine intelligence.

[30] Antoine Maillard,Afonso S. Bandeira,David Belius,Ivan Dokmanić & Shuta Nakajima. (2025). Injectivity of ReLU networks: Perspectives from statistical physics. Applied and Computational Harmonic Analysis101736-101736.

[31] Zhonge Su,Guoqiang Zheng,Guodong Wang,Miaosen Hu & Lingrui Kong. (2025). An IDBO-optimized CNN-BiLSTM model for load forecasting in regional integrated energy systems. Computers and Electrical Engineering(PA),110013-110013.