EAI Endorsed Transactions
on Scalable Information Systems

Research Article EAI.EU

# EBTM: Web Vulnerability Attack Behavior Recognition Method Based on Feature Fusion

Xiangnan Lin, Bin Lu*

Key Laboratory of Cyberspace Security, Ministry of Education, Zhengzhou 450000, Henan, China

## Abstract

INTRODUCTION: With the continuous expansion of network scale, effectively detecting web vulnerability attacks has become a critical challenge for ensuring network security. Existing research, primarily focused on balanced sample recognition, proves inadequate for real-world scenarios due to high false alarm rates in large-scale traffic and poor recognition performance for minority attack samples caused by imbalanced data distribution.
OBJECTIVES: This paper aims to address the limitations of current web vulnerability attack detection methods in imbalanced real-world environments. The objective is to propose a novel recognition method that reduces the false positive rate and improves the identification performance for minority attack samples under highly skewed data distributions.
METHODS: We propose a feature fusion-based recognition method named EBTM for imbalanced samples. The method integrates expert knowledge to optimize feature selection, focusing on key information and ensuring a more uniform mapping of URL requests. It employs three output features from different advantageous models for feature fusion, thereby generating a richer and more discriminative feature representation for the final recognition task.
RESULTS: Experimental results demonstrate that the proposed EBTM method significantly enhances the recognition of web vulnerability attack behaviors. Under a realistic imbalanced condition where attack samples constitute only about 3% of the data, the model achieves a macro-average F1 score of 99.1% and reduces the false positive rate to 0.054%.
CONCLUSION: The EBTM method effectively improves the efficiency and accuracy of web vulnerability attack behavior recognition in practical, imbalanced scenarios. By combining expert-guided feature optimization and multi-model feature fusion, it successfully addresses key challenges of high false alarms and poor minority class recognition, offering a robust solution for securing large-scale network environments.

**Keywords:** Vulnerability Attack Behavior,  Large-Scale Traffic, False Positive Rate, Feature Fusion, Deep Learning.

Received on 15 November 2025, accepted on 15 January 2026 published on 21 January 2026

doi: 10.4108/eetsis.11636

## 1. Introduction

The rapid expansion of the Internet has made HTTP traffic indispensable for web applications and browsing, facilitating vast data transmission and interaction. However, with the surge in HTTP traffic, malicious traffic has also increased significantly, particularly web vulnerability attacks, which have become a major cybersecurity threat. Common attack vectors—such as SQL injection, cross-site scripting (XSS), and distributed denial-of-service (DDoS) attacks—not only lead to sensitive data breaches and system integrity compromises but may also trigger large-scale service disruptions, causing substantial financial losses and privacy risks for businesses and users. Due to their high stealthiness, low technical barriers, and frequent disguise as legitimate HTTP traffic, these attacks often evade traditional security mechanisms, making detection and mitigation challenging.

In recent years, research on web vulnerability attack detection has continued to evolve. Conventional approaches, such as parameterized input validation and filtering [1], rely

*Corresponding author. Email: lubin0371@126.com on allowlists and regular expression matching to restrict and sanitize user inputs. However, these methods are constrained by predefined rule coverage, failing to detect novel attack techniques. Moreover, their effectiveness heavily depends on developers' security expertise, limiting their robustness against sophisticated attacks. With advancements in artificial intelligence, machine learning (ML) models (e.g., logistic regression, random forests, support vector machines) [2–4], deep learning (DL) models (e.g., CNNs, RNNs, LSTMs, GRUs, DBNs) [5–8], and transformer-based pretrained language models [9–12] have been increasingly adopted for attack identification. These methods autonomously learn features from raw payload data, significantly reducing reliance on manual feature engineering and expert knowledge. Nevertheless, existing approaches still suffer from the aforementioned limitations. Table 1 summarizes the differences and constraints of current web attack detection methods regarding sample conditions, decision strategies, data utilization, and false positive rate (FPR) mitigation.

Table 1. Comparison of limitations in web vulnerability attack behavior identification methods

| Method | Dataset | Decision Method | Data Balance | Research on FPR |
|---|---|---|---|---|
| Nguyen et al.[14] | ECML PKDD<br>CISIC 2010 | Single model classification decision<br>Combine harvester | No | No |
| Vartouni et al.[15] | CISIC 2010 | Single model classification decision<br>Random tree partitioning | No | No |
| Luo et al.[13] | CSIC 2010 | Feature fusion of different DL layers | No | No |
| Zhang et al.[16] | CSIC 2010 | Single model classification decision | No | Yes<br>(1.37%) |
| Tian et al.[17] | Private Dataset<br>(Balanced) | Single model classification decision | No | Yes<br>(3.21%) |
| Althubiti et al. [18] | CSIC 2010 | Single model classification decision | No | No |
| BL-IDS[19] | CSIC 2010 | Single model classification decision | No | Yes<br>(1.4%) |
| Zhu et al.[8] | CSIC 2010 | Feature fusion of different models | No | Yes<br>(0.43%) |
| Odumuyiwa et al.[20] | ECML PKDD<br>CSIC 2010<br>Hybrid 2020 | Single model classification decision | No | Yes |
| ATPAD[21] | CIC-IDS-2017<br>CSIC 2010 | Single model classification decision | No | No |
| Our Method | Private Dataset<br>(Unbalanced) | Multi level model feature fusion | Yes | Yes |

As shown in the table, while many methods achieve high classification accuracy in specific tasks, they often overlook data imbalance and false positive control—two critical challenges in real-world large-scale traffic environments. The imbalanced distribution of benign and malicious traffic severely impacts classifier performance, and minimizing FPR for benign traffic is essential for practical deployment. Furthermore, most methods rely on single-model architectures for feature extraction and classification. Although some studies [8, 13] employ multi-model or multi-layer fusion strategies, they still lack hierarchical textual analysis, failing to fully exploit the potential of annotated datasets. Recent techniques leverage generative adversarial networks (GANs) and diffusion models for data augmentation and synthesis, yet the inherent uncertainty of generative algorithms may compromise model reliability. Thus, the key challenge lies in integrating expert knowledge to extract more discriminative features while ensuring classifier robustness and efficiency in imbalanced, high-volume traffic scenarios.

Although existing methods have made progress in identifying web vulnerability attacks, there are still critical issues to be addressed: first, the false positive rate in large-scale traffic environments cannot meet practical requirements; second, the imbalanced distribution of benign and malicious samples leads to poor identification performance for minority-class (attack) samples. Specifically, a high false positive rate not only increases the workload of security analysts but may also compromise the effectiveness of defense strategies. Due to imbalanced data distribution, current identification models often face performance bottlenecks when detecting minority-class (malicious) samples, further exacerbating the false positive issue. Therefore, improving the model's ability to identify malicious samples has become a key challenge.

To address this problem, this paper proposes EBTM, a feature fusion-based method for identifying web vulnerability attacks. By standardizing irrelevant information in text and enhancing the model's ability to learn key features, combined with a multi-level feature fusion approach, we construct complementary base models at the word, sentence, and context level to improve identification accuracy in imbalanced data environments and reduce the false positive rate in large-scale traffic scenarios.

The main contributions are as follows:

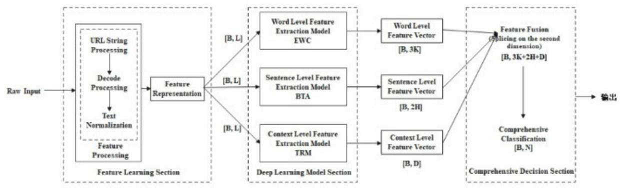
1. To minimize interference from irrelevant information, this study focuses on key features and incorporates expert knowledge to map URL requests into a relatively unified representation, thereby strengthening the model's ability to learn important characteristics.
2. By combining a multi-level feature fusion method, we build complementary feature extraction models at the word,

sentence, and context levels. Through hierarchical feature enhancement, the model gains a more comprehensive understanding of samples, improving its ability to identify minority-class samples

## 2. Method

### 2.1 Overall Framework

The EBTM methodology comprises three core components: the feature learning module, the deep learning model module, and the integrated decision-making module. As illustrated in Fig. 1, the workflow proceeds as follows:



Raw Input
URL String Processing
Decode Processing
Text Normalization
Feature Processing
Feature Representation
Feature Learning Section
[B, L]
[B, L]
[B, L]
Word Level Feature Extraction Model EWC
Sentence Level Feature Extraction Model BTA
Context Level Feature Extraction Model TRM
Deep Learning Model Section
Word Level Feature Vector
[B, 3K]
Sentence Level Feature Vector
[B, 2H]
Context Level Feature Vector
[B, D]
Feature Fusion (Splicing on the second dimension)
[B, 3K+2H+D]
Comprehensive Classification
[B, N]
Comprehensive Decision Section
输出

**Figure 1.** EBTM architecture

The input text information is first sent to the feature learning section for feature processing, which normalizes the URL string and maps it to a relatively uniform expression to reduce the impact of irrelevant information on feature representation and model performance. Secondly, the data that has undergone data preprocessing steps such as decoding and normalization is characterized and processed into feature vectors, which are then fed into various deep learning models to extract feature vectors at different levels using the characteristics of different models. After obtaining the feature vectors of each level, they are sent to the comprehensive decision-making section. The feature vectors of the three levels are concatenated in the second dimension to achieve the fusion of features at different levels. Finally, they are sent to the classifier for classification processing to obtain the final recognition result. The detailed dimensional flow between each module is shown in the characters in Fig. 1, and the correspondence between characters and specific meanings is detailed in Table 2.

Table 2. Symbol definition correspondence table

| Symbol | Corresponding Parameters | Meaning Explanation |
|---|---|---|
| *B* | batch_size | Batch size |
| *L* | max_len | Input sequence length |
| *V* | vocab_size | Character table size |
| *K* | num_kernels | Number of convolution kernels |
| *E* | embedding_dim | Word embedding dimension |
| *H* | hidden_dim | LSTM hidden layer dimension |
| *D* | d_model | Dimension of Transformer Model |
| *M* | memoryslot_num | Number of memory units |
| *N* | num_classes | Number of classification categories |

### 2.2 Feature Learning Section

The EBTM methodology comprises three core components: the feature learning module, the deep learning model module, and the integrated decision-making mod-ule. As illustrated in Fig. 1, the workflow proceeds as follows:

Feature learning serves as the first critical component of the EBTM method for web vulnerability attack behavior identification, playing a pivotal role in effective information extraction throughout the deep learning-based classification process. The quality of feature learning fundamentally determines the upper limit of identification capability. Our feature learning framework consists of two key steps: feature pro-cessing and feature representation. This section first addresses the feature processing of input text information. By focusing on key information and incorporating expert knowledge, we obtain task-relevant feature data. Subsequently, the processed features undergo feature representation. The complete data processing flow of the fea-ture learning module is illustrated in Fig. 2. The following sections will elaborate in detail on the methodologies for feature processing and representation:

#### Feature Information Processing

*URL String Processing*

In diverse environments encompassing various systems, platforms, and servers, URL requests exhibit multiple formats due to configuration differences. Furthermore, when executing attacks, attackers often deliberately craft URL variants to obfuscate their structures and parameters. To more effectively identify web attacks concealed in different URL request formats, this paper proposes a transformation vocabulary that maps URL requests to a relatively unified representation. This approach reduces interference from irrelevant information on model performance, decreases the complexity of the identification process, and enhances the model's capability to recognize key attack-type features. Specifically, we define a transformation vocabulary as shown in Table 3. Strings unrelated to attack types - such as pure numerical information, file type information, and other fixed data - are matched through purpose-built regular expressions and replaced with predefined category identifiers. This focuses the model on attack-relevant feature information within URL requests. It is important to note that all symbols are preserved to avoid losing structural, semantic, or other critical features, ensuring the model can capture potential attack information. Below is an example of a collected URL request that has undergone this unified representation processing:

/C6/JHSoft.Web.WorkFlat/DBModules.aspx/?interfaceID=1;WAITFOR+DELAY+'0:0:5'-- HTTP/1.1

It is converted into a new expression as follows:

/Path/JHSoft.Web.WorkFlat/DBModulesFiletype/?Session&ID=Number;WAITFOR+DELAY+'Number'-- HTTP/Number

Table 3. Convert word list

| Regular Expression Matching Target | Example |
|---|---|
| All pure numbers and strings composed of numbers and "." ":" in the URL | Number |
| .html,.php,.jsp,.asp,.json,.xml,.css,.js,.txt,.csv,.jpg,.png,.gif,.pdf … | Filetype |
| debug,log,trace,error,info,warn… | Debugging |
| timestamp,date,time,expires,created,updated… | Time&Date |
| sessionId, userId, auth, login, logout, register, password, email… | Session&ID |
| config, settings, preferences, options, admin, manage, dashboard | Config |
| The string composed of a-z and - in the path part of the URL | Path |
| MD5 string | MD5 |

*Decoding Processing*

In response to increasingly sophisticated network security protections, attackers often employ various encoding techniques to bypass security mechanisms. The obfuscation methods include but are not limited to Base64 encoding, URL encoding, and HTML entity encoding, which are designed to conceal malicious code or data during transmission, making it more difficult for detection systems to identify them. By utilizing these encoding methods, attackers make their malicious activities harder to detect through conventional security measures, thereby enhancing the stealth and success rate of attacks. For the extracted data, decoding and restoration are required based on different encoding schemes. This section designs and implements a multi-layer text decoder specifically for common encoding methods, aiming to progressively decode complex texts containing multiple encoding formats. The decoder sequentially processes HTML entity encoding, single and double URL encoding, Unicode encoding, Base64 encoding, Java syntax encoding, and hexadecimal (Hex) encoding. When valid text encoding is detected, corresponding decoding operations are performed to enable batch processing of samples. This section uses SQL injection attack statements as an example to demonstrate the encoding methods processed in this study and their corresponding text before and after decoding, as detailed in Table 4.

Table 4. Encoding method and examples (taking SQL injection vulnerability as an example)

| Encoding Method | Before Mecoding | After Decoding |
|---|---|---|
| URL encoding | (SELECT%20639%20FROM%20PG_SLEEP(15))--.filename.filetype | (SELECT 639 FROM PG_SLEEP(15))--.filename.filetype |
| Unicode encoding | s\u0065\u006C\u0065\u0063\u0074\u0020\u0066\u0072\u006F\u006D | select from |
| Base64 encoding | c2VsZWN0ICogZnJvbSB1c2VycyB3aGVyZSBpZCA9IDEgb3IgMSMiICggdW5pb24gc2VsZWN0IDEsdmVyc2lvbiAoICkgLS0gMQ== | select * from users where id = 1 or 1#" ( union select 1,version ( ) -- 1 |
| HTML entity encoding | admin&#39; or 1 = 1&#35;,1 | admin' or 1 = 1#,1 |
| Hex encoding | 61 64 6D 69 6E 20 29 20 6F 72 20 28 20 31 20 3D 20 31 20 29 | admin" ) or ( "1" = "1 |
| Java syntax encoding | admin\\\" or 1=1 | admin" or 1=1 |
| Two URL encodings | admin%2527or%25201%253D1%2523 | admin'or 1=1# |

*Text Normalization*

To enhance the model's feature extraction capability for web vulnerability attack behavior samples and ensure input consistency, this study performs normalization processing on the decoded and restored text strings. The normalization process comprises five key aspects as follows:

First, case unification is implemented by converting all uppercase English letters to lowercase, avoiding semantic duplication caused by case variations and improving text processing efficiency and accuracy. Second, full-width characters are uniformly converted to half-width characters to standardize text formats and prevent processing errors due to inconsistent character formats. Third, for SQL-based attack payloads, common MySQL comment information is removed by eliminating both single-line and multi-line MySQL comment content, thereby avoiding interference from meaningless characters such as author names and version numbers in attack detection. Fourth, rigorous validation and cleaning of input information is performed, including removal of redundant whitespace characters to prevent attackers from exploiting space insertion for obfuscation bypass, while simultaneously avoiding processing errors caused by excessive spaces to improve text processing robustness. Fifth, unified formatting is applied to date, time and numerical information appearing in the text to enhance feature stability and processing consistency.

## Feature Information Representation

In the feature information representation section, a character level text encoding strategy is adopted to convert the original text sequence into a fixed length, structured numerical tensor, in order to achieve structured expression of information and unified representation of features.

This strategy is based on a predefined character set $C = \{c_1, c_2, ..., c_n\}$ and constructs a mapping relationship between characters and indexes:

$$f_{\text{char2idx}} : C \rightarrow \mathbb{N}^{+} \tag{1}$$

For any input text sequence $T = \{t_1, t_2, ..., t_L\}$, scan character by character and perform a lookup mapping operation to obtain the corresponding index sequence:

$$X = \{f_{\text{char2idx}}(t_i) \mid t_i \in T, t_i \in C\} \tag{2}$$

To achieve uniform tensor input dimensions, we perform standardized processing on the index sequence $X$ through zero-padding and truncation operations to enforce a consistent text length according to the predefined threshold max_len. When the sequence length is insufficient, zero-value vectors are appended to the end until reaching the predetermined length; when exceeding the limit, the trailing portion is discarded while preserving the leading valid characters. Finally, the feature vectors and their corresponding category labels are jointly packaged into PyTorch-format data tensors, providing the required input structure for subsequent deep learning frameworks. This encoding strategy not only effectively preserves fine-grained textual information but also avoids potential errors introduced by certain tokenization methods, thereby delivering high-quality input representations for subsequent feature learning and model training.

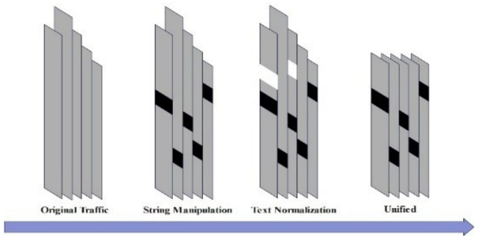


**Figure 2.** The data processing process of the feature learning part

## 2.3 Deep Learning Model Section

Deep learning achieves multi granularity feature extraction by designing deep learning models at the word level, sentence level, and context level. The architecture of the deep learning model is shown in Fig. 2. The following will specifically introduce the deep learning models for feature extraction at each level:

### Feature Information Processing

To effectively extract word level features, this paper constructs an Enhanced WordCNN (EWC) feature extraction model based on multi-scale convolutional neural networks. This model fully utilizes the advantages of convolutional neural networks in local feature extraction, adopting a parallel multi branch convolution structure to achieve efficient representation of word level features.

Specifically, for a given input text $X = \{x_1, x_2, ..., x_n\}$, the first step is to perform embedding mapping:

$$E = \text{Embedding}(X) \in \mathbb{R}^{n \times d} \tag{3}$$

Among them, is the embedding dimension and is the number of characters.

Then, different sizes of one-dimensional convolution kernels are used for feature extraction, and the feature extraction formula is as follows:

$$C_k = \text{GELU}(\text{Conv1D}(E, W_k) + b_k) \tag{4}$$

Among them, $W_k$ is the convolution kernel weight, $b_k$ is the bias term, and $k$ represents different convolution kernel widths, corresponding to different n-gram level features.

GELU improves the stability of gradient propagation through a nonlinear smoothing transformation mechanism, and then applies adaptive max pooling for feature aggregation. Different convolution outputs are concatenated along the channel dimension, and some neurons are randomly suppressed through Dropout to enhance the model's generalization ability. Finally, the model outputs a $3K$ dimensional (see Table 2 for details) feature vector, effectively achieving word level feature extraction in the task of identifying web vulnerability attack types.

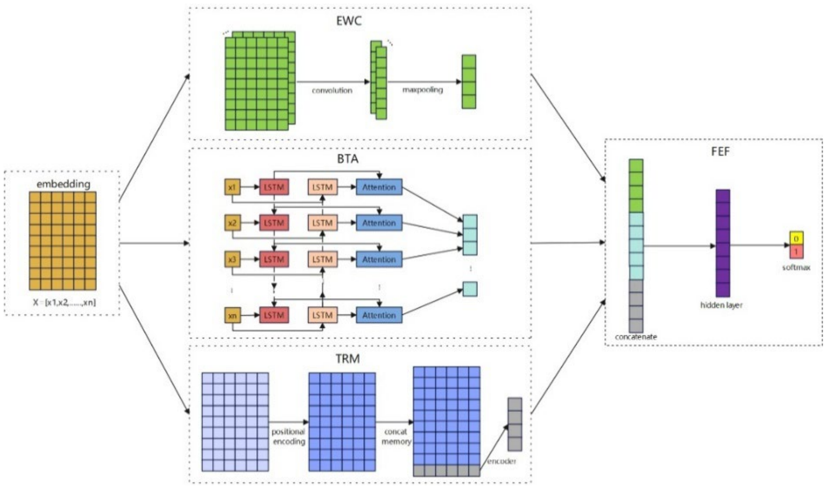


**Figure 3.** EBTM deep learning model Architectures

## Sentence Level Feature Extraction Model BTA

In the sentence level feature extraction deep learning model BTA (BiLSTMAttention), BiLSTM is used to capture global dependencies at the sentence level. BiLSTM captures the contextual dependencies of each word through bidirectional information flow and generates word representations that contain contextual information. This ability is crucial for understanding the semantics of web attack sequences, as attack patterns often rely on specific word sequences and their semantic combination relationships. BiLSTM not only focuses on local word combinations, but also understands the semantic structure of the entire sentence.

Based on the output of BiLSTM, the model introduces an attention mechanism to calculate the attention weights for each time step, in order to focus on the key parts of the sentence. This mechanism not only captures global dependencies, but also effectively enhances the model's perception of attack semantic features while maintaining computational efficiency. The specific calculation formula is as follows:

$$\alpha_t = \frac{\exp(W_a h_t)}{\sum_{j=1}^{n} \exp(W_a h_j)} \tag{5}$$

Among them, $W_a$ is the trainable attention weight matrix, $\alpha_t$ represents the importance of the t-th time step, and $h_t$ is the hidden state of each time step represented by the concatenation of forward and backward LSTM.

Then, the output of BiLSTM is weighted and summed using attention weights, and the concatenated features are regularized through a Dropout layer to prevent overfitting, generating a sentence level global feature vector. The generation formula is as follows:

$$F_{\text{BTA}} = \text{Dropout}(\sum_{t=1}^{n} \alpha_t h_t) \tag{6}$$

The BTA model combines BiLSTM and attention mechanism to achieve efficient extraction and semantic understanding of sentence level features, making it more suitable for analyzing complex text data such as web attack sequences.

## Context Level Feature Extraction Model BTA

The TRM (Transformer_Memory) model is based on the encoder part of the Transformer architecture, using self attention mechanism to represent the relationships between different positions in the sequence, and introducing a memory module on top of it to better extract contextual feature information.

Specifically, after embedding is completed, the model injects positional information into the word embedding vector through positional encoding, enabling the model to capture the positional information of words in the sentence and better understand the contextual relationship of the entire text. At the same time, a learnable memory parameter is introduced to concatenate the word embedding vector injected with positional information, and the concatenated vector is used as the input of the Transformer encoder. The corresponding generation formula is as follows:

$$E = [M; X + \text{PositionalEncoding}] \tag{7}$$

Among them, $M$ is a learnable global memory parameter.

As the Transformer encoder performs self attention calculation on the input vector, the memory vector can interact with each token in the word embedding vector. During the training process, it updates with the gradient and gradually learns how to store global information that is effective for the task. The specific calculation formula is as follows:

$$A = \text{softmax}(\frac{(W_Q E)(W_K E)^T}{\sqrt{d}}) E W_V \tag{8}$$

Among them, $W_Q$, $W_K$, and $W_V$ are trainable parameter matrices.

Finally, the model output is extracted as sequence features, with special attention paid to the feature representation of the first position. The feature vector of this position is processed by the encoder to better integrate global contextual information, which can serve as a semantic summary of the entire input sequence and be fed into the feature fusion model to provide global information.

## Comprehensive Decision Section

In the comprehensive decision section, the adaptive weight adjustment method FEF (Feature Fusion) is designed to fuse features of different granularities. Introducing a gating fusion mechanism to globally scale features, the EWC model, BTA model, and TRM model are responsible for extracting feature representations at different levels from input data. The output feature dimensions are [B, 3K], [B, 2H], and [B, D], respectively. After obtaining features at different levels, the gating feature fusion is used to calculate the importance weight of each feature vector. The specific calculation formula is as follows:

$$g_{\text{EWC}} = \sigma(W_{\text{EWC}} \cdot F_{\text{EWC}}) \tag{9}$$

$$g_{\text{BTA}} = \sigma(W_{\text{BTA}} \cdot F_{\text{BTA}}) \tag{10}$$

$$g_{\text{TRM}} = \sigma(W_{\text{TRM}} \cdot F_{\text{TRM}}) \tag{11}$$

Among them, $\sigma$ is the Sigmoid function, $W_{\text{EWC}}$, $W_{\text{BTA}}$, $W_{\text{TRM}}$ are learnable parameter matrices, $F_{\text{EWC}}$, $F_{\text{BTA}}$, $F_{\text{TRM}}$ are the eigenvectors output by EWC, BTA, and TRM models, respectively.

Normalized by Sigmoid, used for weighted features, the specific formula is as follows:

$$F'_{\text{EWC}} = g_{\text{EWC}} \cdot F_{\text{EWC}} \tag{12}$$

$$F'_{\mathrm{BTA}} = g_{\mathrm{BTA}} \cdot F_{\mathrm{BTA}} \tag{13}$$

$$F'_{\mathrm{TRM}} = g_{\mathrm{TRM}} \cdot F_{\mathrm{TRM}} \tag{14}$$

By using the above method to adaptively adjust the weights of different features, the fusion effect can be improved. Finally, concatenate along the feature dimension (dim=1) to form a comprehensive feature vector [B, 3K+2H+D]. In order to map the fused feature vectors to the target category space, the model uses an MLP as a classifier to complete the classification task.

## 3. Experimental Setup

To better evaluate and validate the performance and effectiveness of the method, this paper conducts the following experimental settings, including the dataset, experimental environment, evaluation indicators, and model parameter settings.

### 3.1 Dataset

The publicly available dataset adopts the widely recognized CSIS-2010, while the self built dataset mainly consists of (1) vulnerabilities directly collected and replicated from public vulnerability databases such as CVE vulnerability database, CNVD database, and exploit db between 2018 and 2023 (2) Competition data for big data security analysis such as DataMen (3) The dataset provided by the cybersecurity competition (4) Organize data from datasets of HttpParams, CIC-IDS 2017, GitHub website, and Kaggle website. Due to the fact that the baseline dataset CSIC 2010 used in most previous experiments [8, 13, 18] cannot cover the data of new vulnerability attack behaviors, it is difficult to accurately evaluate the recognition ability of the model for current mainstream attacks and their variants. Therefore, it is considered to merge and organize the CSIC 2010 dataset with relevant data from recent years to obtain a self-built dataset for future use, which contains a total of 36892 web vulnerability attack behaviors. By preprocessing the relevant data separately, standardizing the format and integrating the feature space. Although there are differences in nor-mal behavior patterns, attack feature distributions, and traffic baselines among data from different sources and periods, the benefits of increasing timeliness and coverage through data integration far outweigh controllable adverse effects such as feature distribution shifts and differences in attack manifestations. To approach the distribution characteristics dominated by normal traffic in real scenarios, 1143584 normal traffic were sorted out, and the experimental construction showed that the ratio of normal traffic to web vulnerability attack traffic was greater than 30:1.

### 3.2 Experimental Environment

The experiment was conducted on a server configured with 60GB of memory, 96 core AMD EPYC 9654 CPU, and NVIDIA GeForce RTX 4090 (24GB of video memory). The server is running Ubuntu 20.04 operating system, and the experiment is based on Pytorch implementation. This chapter randomly divides the integrated dataset into three independent sets, with 80% of the samples used for training and adjusting bias, 10% of the samples used for validation during the training phase, and the final 10% of the samples used for testing the model.

### 3.3 Evaluation Indicators

In order to better evaluate the classification performance of the EBTM model on imbalanced datasets, this chapter uses four different performance evaluation metrics:

Macro Average Precision measures the accuracy of a model in predicting positive cases for each category.

$$\mathit{Macro\text{-}Precision} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FP_i} \tag{15}$$

Macro Average Recall measures the model's ability to identify positive examples for each category.

$$\mathit{Macro\text{-}Recall} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i} \tag{16}$$

Macro Average F1 Score takes into account the balance between precision and recall for each category.

$$\mathit{Macro\text{-}F1\ Score} = \frac{1}{N}\sum_{i=1}^{N}\left(2\cdot\frac{\frac{TP_i}{TP_i+FP_i}\cdot\frac{TP_i}{TP_i+FN_i}}{\frac{TP_i}{TP_i+FP_i}+\frac{TP_i}{TP_i+FN_i}}\right) \tag{17}$$

The False Positive Rate reflects the misjudgement of positive cases in application scenarios by the model.

$$FPR = \frac{FP}{FP + TN} \tag{18}$$

Accuracy is a global score that quantifies the correctness of a model's performance across all categories, but it is bound to achieve good results under extremely imbalanced sample conditions, and the accuracy of various methods will generally be high. Because in imbalanced data, the model can achieve higher accuracy by predicting the majority of samples as the majority class, so accuracy is not considered to evaluate the performance of the model

### 3.4 Model Parameter Setting

The model training adopts the AdamW optimizer, with an initial learning rate of 1e-3, and the loss function uses a cross entropy loss function suitable for classification tasks. The experiment improved the model performance by optimizing some hyperparameter settings, and plotted the loss and macro average F1 score of the training and validation sets as a function of epoch, as shown in Fig. 4. After about 4 rounds, the model loss tended to converge steadily. During the training process, an early stopping mechanism (with a patch set to 5) was introduced to prevent overfit-ting. The batch size was fixed at 32, and the number of training epochs was 30. The experimental results were taken as the average of three runs.

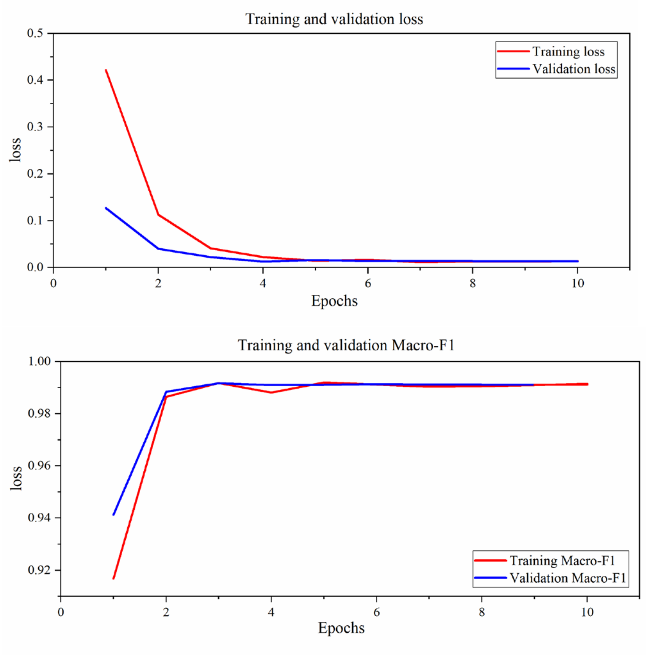


**Figure 4.** Loss/Macro Average F1 Score V.S. Training Round Chart

## 4. Experimental Results and Analysis

To evaluate the performance of EBTM, this section discusses and analyzes its effectiveness in identifying web vulnerability attack behaviors from two aspects: performance comparison and ablation experiments, through a systematic experimental design.

### 4.1 Performance Comparison

In order to comprehensively evaluate the performance of the EBTM model, this method selected five baseline models for comparison, including:

*Zhu et al. [8]:* In the preprocessing stage, this model improves the TF-IDF algorithm by introducing weighting factors to assign higher weights to data with attack content. Then, based on the improved algorithm, a dictionary is constructed to filter low-frequency information in the payload. Word2Vec is used to convert vocabulary into feature vectors, and TextCNN and BiLSTM Attention models are used to extract features and fuse them to determine the type of HTTP request.

*Tekerek et al. [22]:* This model preprocesses and truncates HTTP request headers, segments URLs and payloads, and converts them into two-dimensional matrices, simplifying the complexity of input data. Then, it uses a bag of words model to convert URLs and payloads in HTTP requests into frequency features, preserving character level information while reducing data complexity. It effectively captures common string operations and patterns in web attacks, and finally uses a CNN model for classification tasks.

*BL-IDS [19]:* This model segments the decoded HTTP request according to specific separators (such as "/", "&", "=", "?", etc.) to obtain a series of words or symbols. The Skip Gram model is used to generate word vectors, and the bidirectional long short-term memory network BiLSTM model is used to learn the temporal characteristics of the HTTP request. Finally, the Softmax classifier is used to determine whether the input HTTP request is a normal request or an attack request.

*ATPAD [21]:* This model consists of an encoder RNN, a decoder RNN, and an attention network. The encoder RNN extracts sequence information, the attention mechanism focuses on key parts, and the decoder RNN performs classification prediction to identify abnormal loads in web applications.

*Web FTP [23]:* This model treats web data as natural language sequences and uses the ELECTRA framework to learn deep features of sequence data through a generator discriminator structure. The pre trained model's discriminator is used as the backbone network, combined with fully connected layers and Softmax layers, to fine tune binary classification tasks through real-world traffic data, achieving binary classification detection of web data.

The focus of this section is to compare the four indicators mentioned in section 3.3. As some baseline models are not open source, this section fine tunes them based on the original model design structure, hyperparameter design, and training method to achieve the best results. At the same time, the cost during training is ignored, and the baseline model is trained by maximizing the training epochs to complete the training process when all models converge. The results of the indicators for the self built dataset are shown in Table 5. The experimental results showed that some models had high macro average precision values, while others had relatively high macro average recall values. In this section, we analyzed that in the case of imbalanced samples, the high macro average recall rate proves that the model can better capture positive samples, resulting in low macro average precision because it may misjudge some negative samples as positive; If the macro average accuracy is high, it indicates that the model's predictions are conservative, but some positive samples may be missed. EBTM achieved better performance than other compared models in terms of macro average F1 score and false positive rate, which best reflect the model's attention. This proves the effectiveness and low false positive rate of the model under large-scale imbalanced sample conditions.

Table 5. Comparative experimental results

| Method | Macro Precision | Macro Recall | Macro F1 Score | FPR |
|---|---|---|---|---|
| Zhu et al.[8] | 0.96897 | 0.97516 | 0.97180 | 0.00381 |
| Tekerek et al.[22] | 0.91737 | 0.84196 | 0.87819 | 0.01759 |
| BL-IDS[19] | 0.92470 | 0.94155 | 0.93281 | 0.00917 |
| ATPAD[21] | 0.94330 | 0.89252 | 0.91707 | 0.01193 |
| Web-FTP[23] | 0.96841 | 0.95589 | 0.96225 | 0.00376 |
| EBTM | **0.98352** | **0.99769** | **0.99046** | **0.00054** |

To visually demonstrate the performance of different methods for classification tasks, ROC and PR curves were plotted to evaluate the performance of different methods, as shown in Fig. 5. EBTM and five baseline models were designated as curves 1-6 in the order of introduction. The horizontal axis of the ROC curve represents the false positive rate, and the vertical axis represents the true positive rate. The closer the curve is to the upper left corner, the smaller the area under the curve, indicating better performance of the model. From the experimental results, it can be seen that this method achieves better results compared to the other five methods.

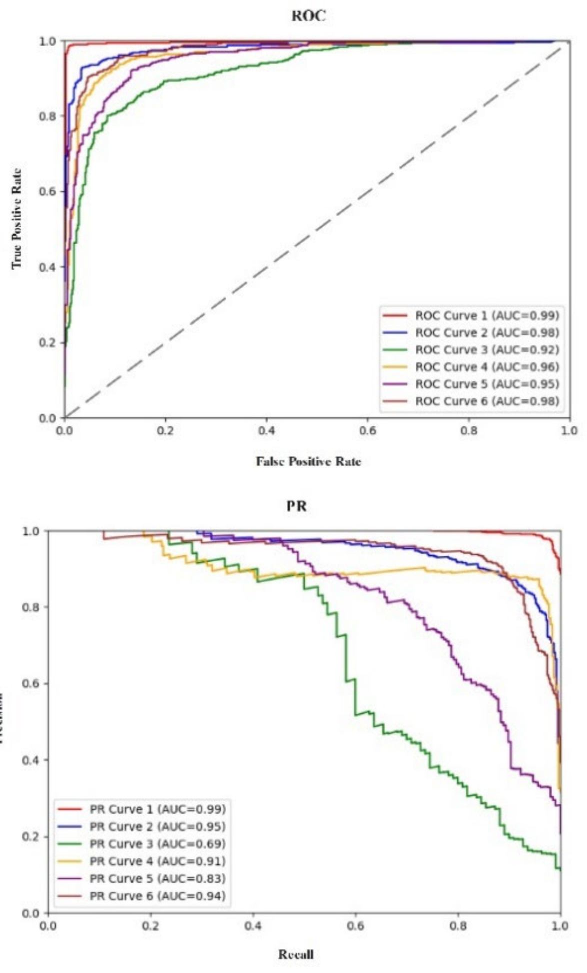


**Figure 5.** ROC and PR curves of different methods

Under the condition of imbalanced samples, the PR curve can better evaluate the performance of the model compared to the ROC curve. The ROC curve focuses on the true positive rate and false positive rate, and has low sensitivity to sample imbalance, while the PR curve combines precision and recall, directly focusing on the performance of positive samples, and can better reflect the true performance under sample imbalance conditions. From the experimental results, it was found that the PR curve of our method is closer to the upper right corner and the offline area is also the largest, which proves the superiority of our method compared to other methods.

## 4.2 Ablation Experiment

In order to further analyze the impact of various parts of EBTM on model performance, four ablation experiments were designed to remove the feature information processing part (FIP), word level feature extraction part (EWC), sentence level feature extraction part (BTA), and context level feature extraction part (TRM) from EBTM. The experimental results are shown in Fig. 6.

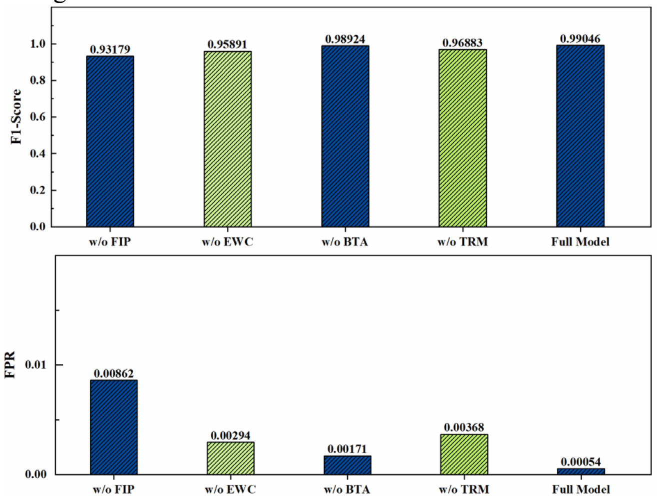


**Figure 6.** Experimental results of EBTM ablation

*w/o FIP:* Without processing feature information, the interference of redundant in-formation makes it difficult for the model to extract feature information well, resulting in a decrease in the model's classification ability. By comparing the classification results of removing FIP module and complete EBTM, it is found that the main reason for the performance decline is the increase in false positives of benign traffic and malicious traffic to varying degrees. This proves that the setting of FIP module plays an important role in optimizing input information and improving model false positives.

*w/o EWC:* Experimental results show that after removing the EWC module, the macro average F1 score decreased by 2.16% and the false positive rate increased by 0.24%. After removing the EWC module, the model lacks capture of word level in-formation, resulting in a decrease in model performance.

*w/o BTA:* After removing the BTA module, the macro average F1 score decreased by 0.12% and the false positive rate increased by 0.12%. Compared to the absence of other feature extraction modules, the impact on model performance was minimal. Prove that the BTA module compensates for some sentence level features that EWC and TRM modules cannot capture, which is helpful for improving the overall model performance.

*w/o TRM:* After removing the TRM module, although the macro average F1 score slightly increased compared to EWC, the false alarm rate did not decrease but instead increased. The analysis of the results shows that compared to the EWC module, the TRM module has a more significant recognition effect on web vulnerability attack behaviors. This phenomenon indicates that the TRM module plays an important role in improving the ability to identify attack behaviors.

The results of the ablation experiment demonstrate that the design of FIP, EWC, BTA, and TRM modules all contribute to the model's recognition ability. When working together, they can fully analyze the input information and achieve optimal model performance. The absence of different modules will have varying degrees and aspects of impact on the model.

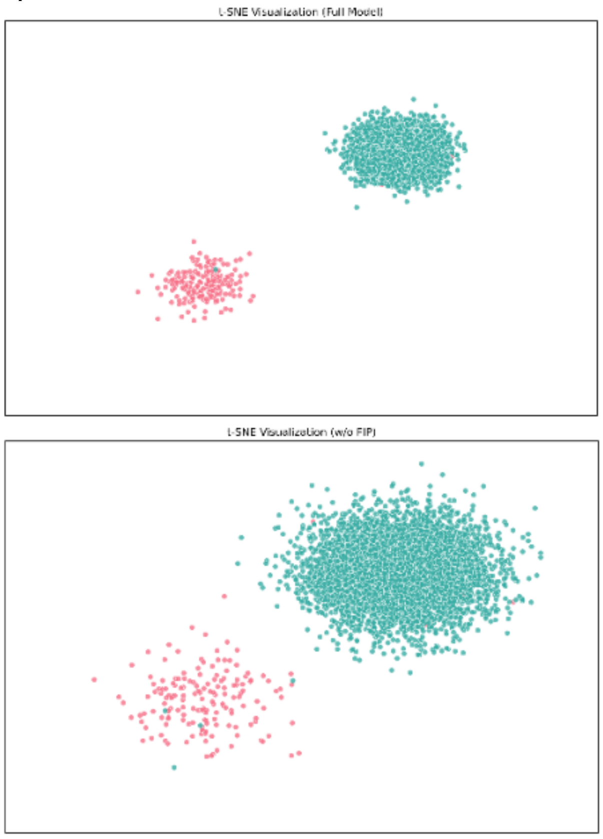


**Figure 7.** T-SNE visualization

In addition, to verify whether the features extracted by EBTM are helpful for recognition performance, t-SNE dimensionality reduction technique was used for visualization analysis, and the results are shown in Fig. 7. The complete model visualization is shown in the left figure. The results indicate that different categories of samples form obvious clustering distributions in the feature space, and only a few samples have classification errors. This result proves that the EBTM model can effectively extract discriminative features under large-scale imbalanced sample conditions through multi granularity feature fusion methods, and improve the classification performance of the model in such scenarios. In addition, this section also presents t-SNE visualization of the model after removing the FIP module, as shown in the figure on the right. Through comparison, it can be found that the design of feature information processing in this model can effectively reduce the impact of irrelevant in-formation on model performance, and has a good effect on aggregating feature in-formation and clarifying the boundaries of clustering regions.

## 4.3 Hyperparameter Sensitivity Analysis

To further understand the impact of hyperparameter settings on the performance of the EBTM model, a sensitivity analysis was conducted, with a focus on analyzing the number of layers in the Transformer encoder.

The experimental results of hyperparameter sensitivity analysis for Transformer encoder layers are shown in Figure 8. The analysis results indicate that when the number of encoder layers is less than 2, the performance of the model improves with the increase of the number of encoder layers. However, when the number of encoder layers reaches 2 or more, the performance of the model tends to stabilize and fluctuates within a very small interval. And the training time increases continuously with the increase of encoder layers, especially when increasing from five to six layers, the training time increases exponentially. Analysis may be due to the non-linear increase in the time complexity of gradient calculation when it needs to go through more layers, especially in the 5-6 layers, where the cumulative effect is more obvious. In summary, selecting a two-layer Transformer encoder can ensure training efficiency while maintaining good performance.

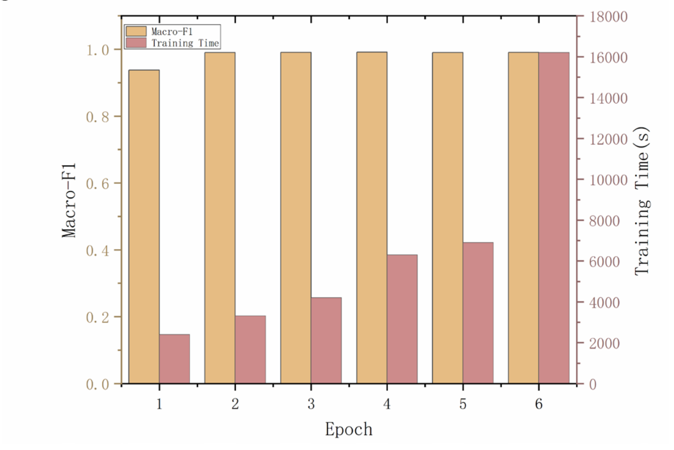


**Figure 8.** Sensitivity analysis of Transformer encoding layers

## 5. Conclusions and Future Work

This chapter proposes a web vulnerability attack behavior recognition method EBTM for large-scale imbalanced network traffic environments, aiming to improve the model's recognition performance of web vulnerability attack behavior in real-world scenarios. This method focuses on key information and incorporates expert knowledge. By mapping URL requests relatively uniformly, it effectively reduces the interference of redundant information on recognition performance and strengthens the model's ability to learn important features. At the same time, EBTM performs multi-level feature extraction on the input text from three granularity levels: word level, sentence level, and context level. It utilizes an adaptive weight feature fusion strategy to achieve complementary enhancement of information at different semantic levels, and ultimately completes classification decisions. The experimental results show that EBTM exhibits better recognition performance compared to existing methods under large-scale imbalanced sample distribution conditions, while reducing false positive rates, and can better adapt to the recognition needs of web vulnerability attack behaviors in actual complex network environments.

In the future, we will continue to study the following topics:

1. The problem of identifying attack types in multi vulnerability fusion scenarios. In actual attacks, different types of vulnerabilities may be compounded and applied to the same attack, forming a more covert and destructive attack chain. However, currently most detection models can only identify such composite attack behaviors as a single vulnerability type, making it difficult to achieve joint identification and accurate classification of multiple vulnerability applications. How to effectively enhance the detection capability of composite attacks will be the focus of the next research step.

2. Further expand the methods for locating and extracting auxiliary attack payloads. In the process of detecting web vulnerability attacks, there is a relative lack of research on extracting attack payloads. Currently, attention weight scores are mainly relied on for auxiliary extraction and analysis. However, for generative large language models, attention weights reflect the degree of attention paid to the input text when generating labels, and it is difficult to directly reflect the impact of the input text on the classification results. It lacks interpretability and can only assist in the localization and extraction of attack payloads. Therefore, future research can further explore more diverse and efficient auxiliary positioning strategies, such as introducing attack payload knowledge graphs to provide context related information, assisting in decision-making and judgment during the extraction process, and further enhancing the practicality and trustworthiness of the system in practical environments.

## References

[1] Sadeghian A, Zamani M, Manaf A A. SQL injection vulnerability general patch using header sanitization [C]//Proceedings of the International Conference on Computer, Communications, and Control Technology (I4CT 2014). Langkawi: IEEE, 2014: 239-42.

[2] Shar L K, Tan H B K. Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns [J]. Information Software Technology, 2013, 55(10): 1767-80.

[3] Kavitha M, Vennila V, Padmapriya G, et al. Prevention of SQL injection attack using unsupervised machine learning approach [J]. International Journal of Aquatic Science, 2021, 12(3): 1413-24.

[4] Anbiya D R, Purwarianti A, Asnar Y. Vulnerability detection in php web application using lexical analysis approach with machine learning [C]//Proceedings of the 5th International Conference on Data and Software Engineering (ICoDSE 2018). Mataram: IEEE, 2018: 1-6.

[5] Alaoui R L, Nfaoui E H. Deep learning for vulnerability and attack detection on web applications: A systematic literature review [J]. Future Internet, 2022, 14(4): 118-39.

[6] Tadhani J R, Vekariya V, Sorathiya V, et al. Securing web applications against XSS and SQLi attacks using a novel deep learning approach [J]. Scientific Reports, 2024, 14(1): 1803-19

[7] Jin X, Cui B, Yang J, et al. Payload-based web attack detection using deep neural network [C]//Proceedings of the 12th International Conference on Broad-Band Wireless Computing, Communication and Applications (BWCCA 2017). Barcelona: Springer, 2018: 482-8

[8] Zhu M, Hong T, Luo Q, et al. A deep learning-based method for HTTP payload classification in attack detection [C]//Proceedings of the Third International Conference on Green Communication, Network, and Internet of Things (CNIoT 2023). Chongqing: SPIE, 2023: 517-23.

[9] Montes N, Betarte G, Martínez R, et al. Web application attacks detection using deep learning [C]//Proceedings of the 25th Iberoamerican Congress(CIARP 2021). Porto: Springer, 2021: 227-36.

[10] Seyyar Y E, Yavuz A G, Ünver H M. An attack detection framework based on BERT and deep learning [J]. IEEE Access, 2022, 10: 68633-44.

[11] Liu Y, Dai Y. Deep Learning in Cybersecurity: A Hybrid BERT−LSTM Network for SQL Injection Attack Detection [J]. IET Information Security, 2024, 2024(1): 5565950-65.

[12] Nana S R, Bassolé D, Guel D, et al. Deep Learning and Web Applications Vulnerabilities Detection: An Approach Based on Large Language Models [J]. International Journal of Advanced Computer Science Applications, 2024, 15(7): 67-83.

[13] Luo C, Su S, Sun Y, et al. A convolution-based system for malicious URLs detection [J]. Computers, Materials Continua, 2020, 62(1): 69-84.

[14] Nguyen H T, Franke K. Adaptive Intrusion Detection System via online machine learning [C]//Proceedings of the 12th international conference on hybrid intelligent systems (HIS 2012): IEEE, 2012: 271-7.

[15] Vartouni A M, Kashi S S, Teshnehlab M. An anomaly detection method to detect web attacks using stacked auto-encoder [C]//Proceedings of the 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS 2018). Kerman: IEEE, 2018: 131-4.

[16] Zhang M, Xu B, Bai S, et al. A deep learning method to detect web attacks using a specially designed CNN [C]//Proceedings of the 24th International Conference on Neural Information Processing (ICONIP 2017) Guangzhou: Springer, 2017: 828-36.
[17] Tian Z, Luo C, Qiu J, et al. A distributed deep learning system for web attack detection on edge devices [J]. IEEE Transactions on Industrial Informatics, 2019, 16(3): 1963-71.
[18] Althubiti S, Nick W, Mason J, et al. Applying long short-term memory recurrent neural network for intrusion detection [C]//Proceedings of the SoutheastCon 2018. Petersburg: IEEE, 2018: 1-5.
[19] Hao S, Long J, Yang Y. Bl-ids: Detecting web attacks using bi-lstm model based on deep learning [C]//Proceedings of the International conference on security and privacy in new computing environments(SPNCE 2019). Guangzhou: Springer, 2019: 551-63.
[20] Odumuyiwa V, Chibueze A. Automatic detection of http injection attacks using convolutional neural network and deep neural network [J]. Journal of Cyber Security Mobility, 2020: 489-514.
[21] Qin Z-Q, Ma X-K, Wang Y-J. Attentional payload anomaly detector for web applications [C]//Proceedings of the 25th International Conference on Neural Information Processing (ICONIP 2018). Siem Reap: Springer, 2018: 588-99.
[22] Tekerek A. A novel architecture for web-based attack detection using convolutional neural network [J]. Computers Security, 2021, 100: 102096-112.
[23] Guo Z, Shang Q, Li X, et al. Web-FTP: A Feature Transferring-Based Pre-Trained Model for Web Attack Detection [J]. IEEE Transactions on Knowledge, 2025, 37(3): 1495-507.