electronics

MDPI

Article

Malicious Cloud Service Traffic Detection Based on Multi-Feature Fusion

Zhouguo Chen [1,2,†], Chen Deng [3,†], Xiang Gao [3], Xinze Li [2,3] and Hangyu Hu [3,*]

[1] School of Computer Science and Engineering, Southeast University, Nanjing 210096, China; czgexcel@163.com
[2] The 30th Research Institute of China Electronics Technology Group Corporation, Chengdu 610041, China; lixinze_30@163.com
[3] School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; dengchen@std.uestc.edu.cn (C.D.); xiangao@std.uestc.edu.cn (X.G.)
* Correspondence: huhy@uestc.edu.cn
† These authors contributed equally to this work.

Abstract: With the rapid growth of cloud computing, malicious attacks targeting cloud services have become increasingly sophisticated and prevalent. To address the limitations of traditional detection methods—such as reliance on single-dimensional features and poor generalization—we propose a novel malicious request detection model based on multi-feature fusion. The model adopts a dual-branch architecture that independently extracts and learns from statistical attributes (e.g., field lengths, entropy) and field attributes (e.g., semantic content of the requested fields). To enhance feature representation, an attention-based fusion mechanism is introduced to dynamically weight and integrate field-level features, while a Gini coefficient-guided random forest algorithm is used to select the most informative statistical features. This design enables the model to capture both structural and semantic characteristics of cloud service traffic. Extensive experiments on the benchmark CSIC 2010 dataset and a real-world labeled cloud service dataset demonstrated that the proposed model significantly outperforms existing approaches in terms of accuracy, precision, recall, and F1 score. These results validate the effectiveness and robustness of our multi-feature fusion approach for detecting malicious requests in cloud environments.

Keywords: cloud service security; malicious traffic detection; multi-feature fusion; attention mechanism; feature selection; dual-branch model

check for updates

Academic Editor: Javid Taheri

Received: 24 April 2025
Revised: 22 May 2025
Accepted: 26 May 2025
Published: 28 May 2025

Citation: Chen, Z.; Deng, C.; Gao, X.; Li, X.; Hu, H. Malicious Cloud Service Traffic Detection Based on Multi-Feature Fusion. *Electronics* **2025**, *14*, 2190. https://doi.org/10.3390/electronics14112190

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the rapid development of cloud computing and security technologies, cloud services have gradually become an indispensable part of the Internet. An increasing number of businesses and data are being deployed on the cloud, making it the preferred choice for many enterprises and individuals to host websites and applications. Software as a Service (SaaS) is one of the cloud-based services that allows providers to reduce maintenance and management costs by shifting traditional on premise deployments to the public cloud. Customers can subscribe to SaaS services based on a per-user payment model. Web applications are deployed on Cloud SaaS platforms and delivered to users through the web, which serves as a medium for daily transactions and business operations. However, as user data is outsourced to the cloud, serious security vulnerabilities are increasing, potentially affecting the reputation of providers and reducing customer subscriptions [1]. Firewall as a Service (FWaaS) is one of the critical components of the cloud computing environment. They need to be able to automatically open and flexibly adjust as needed.

Currently, firewalls in the cloud are primarily based on static security rule configurations or simple rule matching, which limits their flexibility and fails to ensure network security [2].

In recent years, application-layer attacks targeting cloud service environments have significantly increased. A web application hosting service is a common Platform as a Service (PaaS) product that can be used to run and manage web, mobile, and API applications, and attacks against it are particularly prominent. Analysis shows that the vast majority of web application security vulnerabilities stem from insufficient validation of client or environmental inputs. This flaw has led to the proliferation of major security threats such as file inclusion attacks, SQL injection attacks, XSS attacks, and CC attacks (a type of application-layer DDoS attack). These malicious requests may cause hosted cloud services to face risks such as service interruptions, content tampering, and data breaches, resulting in significant losses. In June 2024, Snowflake, a leading cloud-native data warehousing platform, suffered a major breach. Attackers exploited the exposed credentials via SQL injection, compounded by the lack of multi-factor authentication (MFA). The incident compromised the sensitive data of 30 million users across 165 organizations, including high-profile clients like AT&T and Ticketmaster [3].

Although the combination of Web Application Firewalls (WAF) and Firewall as a Service (FWaaS) [4] is widely adopted as a protective solution, their rule-based filtering mechanisms remain rigid and struggle to effectively counter new types of attacks. In contrast, machine learning [5–7] technologies can provide intrusion detection systems with dynamic adaptability, enabling them to more effectively identify and defend against emerging security threats, offering a new approach to cloud security protection.

In current cloud service security technologies, the detection of malicious requests targeting cloud service environments still faces the following issues. Existing methods are based on traffic attributes [4], statistical features [8,9], and payload field features. However, feature extraction is singular, failing to fully utilize these two types of features. Additionally, they do not effectively leverage the rich information contained in cloud service logs, resulting in weak model generalization capabilities. Cloud service logs contain structured statistical features and raw payload field features, which contribute to the generation of high-quality training datasets.

We propose a novel dual-branch malicious request detection model tailored for cloud service environments, which leverages a multi-feature fusion strategy to enhance detection performance and generalization. Specifically, the model is designed with a multi-layer network architecture that independently processes and learns from two complementary feature types: statistical attributes (e.g., field lengths, entropy) and field attributes (e.g., textual content of parameters). To address the limitations of traditional feature concatenation, we introduced an attention-based field feature fusion mechanism that dynamically weights the contribution of different fields according to their relevance to malicious behaviors. For statistical features, a Gini coefficient-guided feature selection process based on a random forest model was employed to remove redundant or irrelevant dimensions, improving model efficiency and robustness. By integrating these features through a dual-branch network, the model captures both semantic and structural characteristics of HTTP request traffic, enabling more accurate and resilient detection of malicious requests across diverse cloud service scenarios.

The remainder of this paper is organized as follows. Section 2 introduces related work, Section 3 describes the overall process, Section 4 discusses the features of log data and their preprocessing methods, Section 5 introduces the deep learning framework, Section 6 evaluates the performance of relevant models based on the experimental results, and Section 7 concludes the work.

2. Relevant Work

Current work on malicious cloud service traffic detection can be divided into three types: methods based on traffic attributes, methods based on statistical attributes, and methods based on field attributes. A brief comparison of these three methods is shown in Table 1.

Table 1. Comparison of malicious traffic detection methods.

Method	Category	Description	Advantages	Disadvantages
Traffic attributes	Rule-based	This method focuses on basic traffic characteristics, including IP addresses and URLs, and builds rules based on them	Simple implementation, low cost, and easy to explain	Poor adaptability to new variants of traffic and easy failure of rules
Statistical attributes	Rule-based, machine learning-based	This method is based on statistical features used to construct rules or models, such as field length, packet size, time interval, etc.	Generalization, high detection accuracy, and resistance to mild confusion	Easily affected by traffic disturbances and lacks contextual understanding ability
Field attributes	Machine learning-based, deep learning-based	This method focuses on field characteristics, namely load content. It uses models to understand the semantic information of requests, such as sequence patterns, context dependencies, etc.	Captures context and deep patterns, adapts to complex attack methods	High feature dimensionality, lack of statistical/structural information, may be overfitted and have weak interpretability

Methods based on traffic attributes analyze network protocols and their fields to extract relevant features and identify anomalies in protocol usage. Denning [10] proposed a network intrusion detection system in 1987, which uses network flow features such as source IP addresses to build a rule library for detection. Meisam Eslahi [11] grouped data based on network flow features such as source IP addresses, destination IP addresses, URLs, User agent strings, and timestamps, applying filters to remove false-positive traffic, thereby reducing the false-positive rate of traffic detection.

Methods based on statistical attributes include the work of Krügel [12], who achieved high detection rates for malicious requests by extracting statistical features such as field length, distribution of special characters, and the number of parameters in request packets. Setiyaji [13] measured the word frequency of attack payloads and used machine learning methods to classify malicious traffic.

In recent years, methods based on field text have become increasingly popular. These methods leverage the textual information of payloads for identification but often overlook some more obvious features in the statistical dimension. Zhang [14] extracted and vectorized features from the URL field of request packets based on expert experience, then trained classification models using machine learning methods. Shahin [15] encoded attack requests using the N-gram word vector model and classified malicious requests based on a decision tree algorithm. Ma [16] represented requests using TF-IDF for word embedding and detected malicious requests using an SVM model. In the field of natural language processing, Kim [17] proposed a Text Convolutional Neural Network (TextCNN), successfully applying convolutional neural networks to text feature extraction. Xie [18]

suggested an attack detection method that uses Elastic Pooling (EP) CNN to automatically extract hidden shared features of attacks and accurately identify attack traffic. Most of these methods only extract features from the URL parameters of request packets, resulting in singular feature extraction and insufficient utilization of features.

To enhance malicious request detection generalization, Wu [19] proposed Robust Transformer-based Intrusion Detection System (RTIDS), a novel multi-class classification framework capable of processing complex high-dimensional data. Momu [20] introduced a self-attention mechanism combined with Long Short-Term Memory (LSTM) for malicious request detection. Their approach involves tokenizing request packets, generating word embeddings, and feeding them into LSTM and self-attention networks for training, with experimental results demonstrating superior detection performance. Furthermore, Bokolo [21] treated HTTP requests as long-text data, leveraging Distilled Bidirectional Encoder Representations from Transformers (DistilBERT) pretraining models to extract URL semantic information. Their method capitalizes on the representational divergence between normal and malicious requests in semantic space to achieve effective detection. Jianping W [22] introduced attention mechanism and constructed a Federated Graph Attention Network (FedGAT) model to evaluate the interaction between nodes in the graph, thereby improving the accuracy of internal network interactions and enhancing the accuracy of attack detection.

Our model combines statistical features and field features, adopts multi-feature fusion technology, and utilizes a dual-branch classification network for feature extraction and classification. By comprehensively considering the field attributes and corresponding statistical attributes in request packets for feature extraction, it avoids the issues of false positives and false negatives caused by the loss of critical feature information.

3. Overall Process

As shown in Figure 1, we propose a malicious request detection model based on multi-feature fusion, which includes two branches: field attributes and statistical attributes. The input data are primarily based on cloud service communication logs, mainly consisting of cloud service request packets.

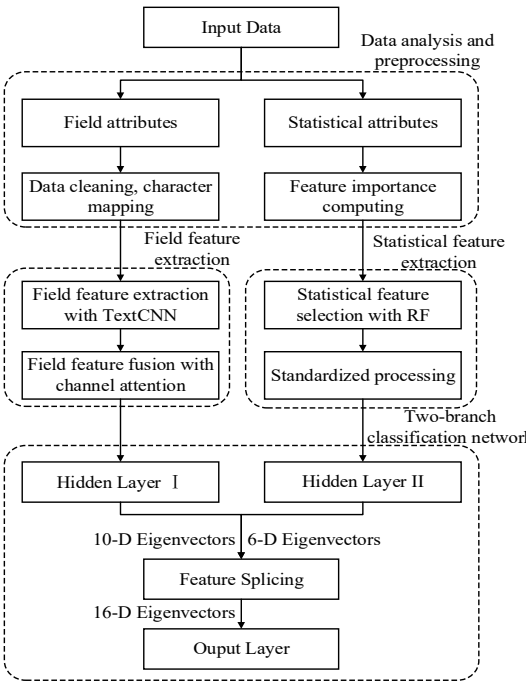

Figure 1. Malicious request detection model framework based on multi-feature fusion.

Attention-based field feature fusion. For the field attributes branch on the left, our model selects multiple fields containing sufficient semantic information, such as transmission parameters, Cookie, User agent, Referrer, and X-forwarded for. The selected field texts are cleaned through steps such as decoding, stop-word removal, and normalization. Subsequently, text mapping techniques such as character encoding and word embedding are used to represent the cleaned data. The TextCNN network is employed to extract field features from each field. Then, a feature fusion method based on a channel attention mechanism is utilized to construct the field fusion feature vector of the request packet.

Gini-based statistical feature selection with Random Forest. For the statistical attributes branch on the right, our model selects multiple field-based statistical attributes, such as length, count, information entropy, and coincidence index. The Gini coefficient is used to evaluate the importance of these attributes. Based on a given threshold, the Random Forest (RF) algorithm is applied to select statistical features, resulting in a corresponding feature subset. Finally, the method standardizes each statistical feature in the subset to construct a statistical feature vector that accurately represents the effective information in malicious requests.

Dual-branch network with specialized feature treatment. Additionally, we designed a multi-layer network classification model with a branch structure to train the field feature vector and statistical feature vector, balancing the contributions of field features and statistical features to the training of the detection model. Unlike traditional single-path structures, this model independently processes semantic and statistical features through two separate branches for enhanced detection accuracy and generalization.

4. Data Analysis and Preprocessing

To facilitate retrieval and management, most cloud services use structured text data to store cloud service logs. These logs not only contain standardized but less distinctive statistical features extracted from packets but also include a large amount of free-text-based field features. How to effectively utilize this information to improve model training performance is a key focus of this chapter.

4.1. Statistical Feature Description

The injection of attack code not only causes malicious request packets to exhibit special patterns in field attributes but also leads to significant differences in statistical attributes such as packet length, number of parameters, and the count of special strings compared to normal request packets. Therefore, these statistical attributes can also serve as the basis for identifying malicious requests. This paper analyzed the statistical attributes of request packets from multiple perspectives and presents the considered statistical properties in Table 2, including length statistical features, quantity attributes, information entropy, and coincidence index. All statistical features in Table 2 were normalized to the [0, 1] range using min–max scaling and then formed into statistical feature vectors.

Length Statistical Features. Due to the embedding of continuous attack code, malicious request packets are typically longer in terms of packet length and related field lengths. In contrast, normal requests are constructed using standard syntax, resulting in relatively shorter field lengths. Therefore, the length attributes of HTTP request packets and their associated fields can be used as a basis for identifying malicious requests. The length features primarily focused on are shown in Table 2 (Features 1–12).

Quantity Statistical Features. Since attack code is often injected into HTTP request packets through multiple transmission parameters or Cookie parameters, the number of parameters in malicious requests is significantly higher than in normal requests. Thus, the number of transmission parameters and Cookie parameters can be used to distinguish

between normal and malicious requests. Additionally, to construct attack code, web attackers often embed a series of special characters in malicious request packets, which rarely appear in normal requests. Therefore, the count of special characters can also be used to identify malicious requests. The statistical features focusing on quantity measurements are listed in Table 2 (Features 13–20).

Table 2. Statistical feature set.

Id	Feature Name	Description
1	http_len	Length of HTTP request
2	url_len	Length of URL field in HTTP requests
3	cookie_len	Length of cookie field in HTTP requests
4	ua_len	Length of user-agent field in HTTP requests
5	referer_len	Length of referrer field in HTTP requests
6	xff_len	Length of the X-forwarded for field in HTTP requests
7	params_len	Length of transmission parameters in HTTP requests
8	params_max_len	Length of the longest transmission parameter in HTTP requests
9	params_mean_len	Average length of transmission parameters in HTTP requests
10	cookie_max_len	Length of the longest cookie parameter in HTTP requests
11	cookie_mean_len	Average length of cookie parameters in HTTP requests
12	params_cnt	The number of parameters transmitted in HTTP requests
13	cookie_cnt	The number of cookie parameters in HTTP requests
14	special_cnt	The number of special characters in HTTP requests
15	url_special_cnt	The number of special characters in URL fields in HTTP requests
16	cookie_special_cnt	The number of special characters in cookie fields in HTTP requests
17	ua_special_cnt	The number of special characters in the user agent field of HTTP requests
18	referer_special_cnt	The number of special characters in the referrer field of HTTP requests
19	xff_special_cnt	The number of special characters in the X-forwarded for field in HTTP requests
20	params_special_cnt	The number of special characters for transmission parameters in HTTP requests
21	entropy	The average amount of information contained in each message received
22	IC	The probability of picking two identical characters at random from a given string

Information Entropy. Information entropy [23] refers to the average amount of information contained in each received message, measuring the disorder and uncertainty of the given information. Its calculation formula is shown in Equation (1). Here, N represents all possible values of a random variable, and p_i is the probability of the i-th random variable occurring. To obfuscate and evade detection systems, malicious request packets often contain a large number of random characters, resulting in higher information entropy compared to normal request packets, which have more fixed content values.

$$\text{entropy} = -\sum_{i=1}^{N} p_i \log(p_i) \tag{1}$$

Coincidence Index. The coincidence index, also known as the probability of coincidence [24], refers to the probability of randomly selecting two identical characters from a given string. Due to the high randomness of characters in malicious request packets, their coincidence index is relatively lower compared to normal request packets [24]. Therefore, the coincidence index can serve as a feature indicator for detecting malicious requests. The mathematical formula for the coincidence index is defined in Equation (2), where N represents the length of the string, n_i denotes the number of times the i character appears in the string, and c represents the total number of distinct characters in the text.

$$IC = \frac{\sum_{i=1}^{c} n_i(n_i - 1)}{N(N-1)} \tag{2}$$

4.2. Field Feature Description

We regard HTTP request messages as a kind of text sequence data, and we obtain text data by extracting field features from the fields of cloud service logs. During the process of field selection, it is necessary to analyze the function and behavior of the field, try to avoid the situations of meaning inclusion and irrelevant fields, and ensure that the selected fields have sufficient semantic information. For example, the string of the transmission parameter field consists of parameter name, parameter value and separator "&", which reflects the behavior of the request and has sufficient semantic information. Another example is that the field containing the information of the backend server is an irrelevant field and needs to be excluded, such as the backend field of protocol, status, port, etc. Under this condition, the selection priority of the transmission parameter field of the request is higher.

Based on the analysis of a large amount of real cloud service log data, we have summarized five field parameters that can be used to detect abnormal traffic, as shown in Table 3, namely transmission parameters, Cookies, User agent, Referrer, and X-forwarded for. Preprocessing and feature extraction will be performed based on these fields in the future.

Table 3. Normal and abnormal values of common fields in cloud service logs.

HTTP Request Field	Normal Value	Abnormal Value
Transmission parameters	user = 72 & group = 32	user = 72 & group = −1 union select 1,2,0,md5(1122)
Cookie	JSESSIONID = 23391DBBADEC19FE; item = phone	JSESSIONID = -admin') union select 1,2,database()#; &item = phone
User agent	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36	<script>alert(1)</script>
Referrer	www.google.com (accessed on 16 January 2025)	/etc/passwd
X-forwarded for	101.99.42.24	-1′ OR 225-25-1 = 0 + 0 + 0 + 1 --,101.99.42.24

4.3. Field Feature Preprocessing

4.3.1. Text Cleaning

Text cleaning transforms the random text within cloud service log fields into standardized text strings.

Decoding. Some attackers disguise attack codes by encoding them [25], converting their string forms to hide their original semantic information, which hinders model learning and training. Common encoding methods in request packets include Base64 encoding, URL encoding, and Hex encoding. This cleaning decodes the encoded content based on whether the field is encoded using the above methods to obtain the original text of the field. As shown in Figure 2, after decoding the URL parameter, 2f6574632f706173737764 is replaced with /etc/passwd, revealing the appearance of a Local File Inclusion (LFI) attack.

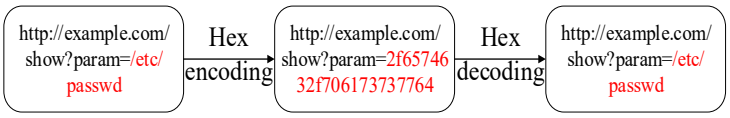

Figure 2. An example of a HEX encoding and decoding process, the emphasizing part is the payload of the encoding and decoding.

Stop Word Removal. Since cloud service logs adhere to a structured storage format and must ensure compatibility, some free-text information may be converted by adding

escape characters. These escape characters are often meaningless to the original data. Stop words, similar to escape characters, not only appear frequently in specific texts but are also ubiquitous in almost all texts, reducing the density of effective information and affecting the generalization ability of the model. Therefore, neutral words that have little impact on text differentiation can be removed. Our paper implemented stop word removal using the nltk package.

Normalization. For fields with insufficient length, we perform zero-padding, and for fields that are significantly too long, we truncate them to ensure uniformity in the length of text strings.

4.3.2. Text Mapping

By performing text mapping on the target text strings, serialized text vectors are generated for model training.

Serialization. The field information in cloud service logs exhibits significant randomness. Character-level text mapping provides finer-grained feature representation compared to word-level text mapping (e.g., word2vec), helping to capture subtle differences while avoiding the issue of unknown words. Additionally, the values of characters in the data are relatively fixed, primarily consisting of uppercase and lowercase letters, numbers, and some special symbols. Therefore, our model uses ASCII code encoding for text serialization as shown in Figure 3.

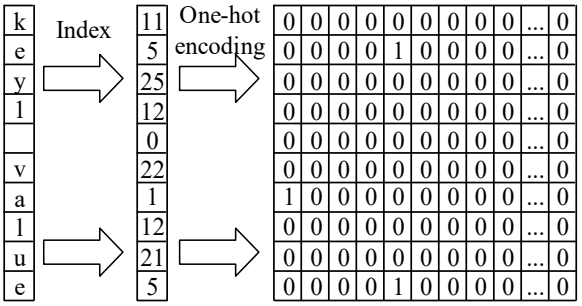

Figure 3. The process of using ASCII code for one hot encoding for text mapping.

Character Embedding. One-hot encoding converts serialized text into text vectors. For a field text X, each field is processed using one-hot encoding as described above, resulting in a corresponding one-hot encoding matrix H of size N $\times$ 95. As shown in Figure 3, this matrix is referred to as the text vector, which can be used as input for training the improved TextCNN model.

Embedding Layer. The one-hot matrix obtained from character encoding is sparse, with most elements being 0, resulting in low space utilization and high storage costs. Additionally, this encoding method cannot capture the relationships between consecutive characters, as the encoding of different characters is independent. To scientifically and efficiently represent the field text of HTTP request packets, it is necessary to use word embedding methods to represent individual characters as low-dimensional, dense character feature vectors. Our model trains an Embedding layer to obtain character embedding vectors for the field text. This allows the distance between feature vectors to represent the degree of correlation between different characters: characters with high correlations have vectors that are closer in the feature space, while those with low correlations are farther apart.

The Embedding layer is a fully connected neural network model specifically designed for text word embedding representation. Its structure is shown in Figure 4. For a given character c, it is first encoded using one-hot encoding to obtain the corresponding character

encoding vector. This vector is then input into the Embedding layer, which performs mapping calculations to convert it into a corresponding dense vector.

$$\underset{\text{encoding vector}}{\underset{\text{One-hot}}{\begin{pmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\end{pmatrix}}}\underset{\text{Weight vector}}{\begin{pmatrix}w_{11}&w_{12}&w_{13}\\w_{21}&w_{22}&w_{23}\\w_{31}&w_{32}&w_{33}\\w_{41}&w_{42}&w_{43}\\w_{51}&w_{52}&w_{53}\\w_{61}&w_{62}&w_{63}\end{pmatrix}}=\underset{\text{Embeded vector}}{\begin{pmatrix}w_{11}&w_{12}&w_{13}\\w_{21}&w_{22}&w_{23}\end{pmatrix}}$$

Figure 4. Embedding layer word embedding diagram.

5. Model Training

5.1. Field Feature Extraction Based on Improved TextCNN

We treat cloud service traffic as a type of text data and used an improved TextCNN to extract field features, obtaining feature vectors for each field. For text tasks, since the input data is an embedding matrix formed by vertically stacking character vectors, the width of the convolutional kernel in the TextCNN model is set to match the dimension of the character vectors to ensure the integrity of the character vectors. The convolutional kernel only slides along the stacking direction of the character vectors, i.e., vertically, to perform convolution operations, thereby extracting the feature vectors of the text.

The model structure of this feature extraction method is shown in Figure 5. The model input is the embedding matrix of each field in the request packet. First, multiple convolutional kernels of different heights are used to perform convolution operations on the character embedding matrix, extracting multiple sets of local features. Then, max-pooling is applied to the results of each convolutional kernel to discard redundant features and retain the optimal local features. Finally, by concatenating the results processed by each pooling layer, the feature vector corresponding to the field is obtained.

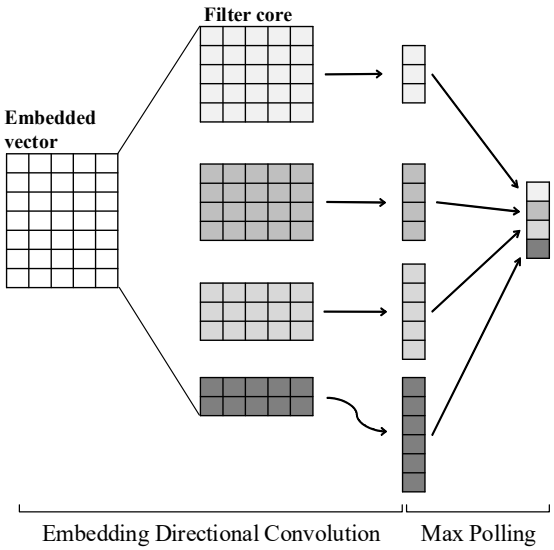

Figure 5. The framework of TextCNN.

Let the embedding matrix of a certain field in the HTTP request be v, which consists of mcharacter vectors, i.e., $v = [v_1, v_2, \ldots, v_m]$. For a sequence of n convolutional kernels $k = [k_1, k_2, \ldots, k_n]$, the width of each convolutional kernel k_i is the same as the width of the character vectors. For the character embedding matrix v, the convolution operation based on k_i can be expressed as the following:

$$h_i = v * k_i \tag{3}$$

where h_i represents the vector obtained by convolving the character embedding matrix with the i convolutional kernel of the convolutional layer and $*$ denotes the convolution operation. By sequentially applying the convolution kernel sequence KK to the character embedding matrix XX, a feature sequence $h = [h_1, h_2, \ldots, h_n]$ can be obtained. For each feature vector in the sequence h_i a max-pooling operation is performed where the maximum value in the vector is taken as the feature extracted by the corresponding convolutional kernel. By repeating this operation for each vector in the feature sequence, the feature vector w corresponding to each field can be obtained as the following:

$$w = [\max(h_1), \max(h_2), \ldots, \max(h_n)] \tag{4}$$

In the improved TextCNN model, to extract features of malicious request packets from different receptive fields, convolutional kernels of sizes 2×95, 3×95, 4×95 and 5×95 are used for convolution operations. Additionally, four convolutional kernels of the same size are selected to extract different features of malicious request packets within the same receptive field. Therefore, the improved TextCNN model employs a total of 16 convolutional kernels to perform convolution on the embedding matrix of each HTTP request field. After max-pooling, the dimensionality of the resulting field feature vector is 16.

5.2. Field Feature Fusion Based on Attention Mechanism

Based on the field feature extraction method described above, the feature vectors corresponding to different fields can be obtained. Before classifying using the aforementioned field feature vectors and statistical feature vectors, these feature vectors need to undergo further fusion operations.

In machine learning methods, different feature vectors are often directly concatenated to obtain a fused feature vector, which means connecting the padded vectors one by one as shown in Figure 6a below. However, since each field feature vector has a different level of importance in characterizing the HTTP request packet, directly concatenating the field feature vectors may result in an inaccurate representation of the request packet. Noisy or irrelevant features (e.g., padding values in fixed-length vectors) may dilute the contribution of discriminative features, distorting the semantic integrity of the request. The model must allocate additional capacity to suppress low-value features, increasing the risk of overfitting (when redundant features dominate) or underfitting (when key features are insufficiently weighted). For example, in SQL injection detection, the payload field's feature vector should dominate the concatenated representation, but direct concatenation treats it equally with less relevant fields like Cache Control.

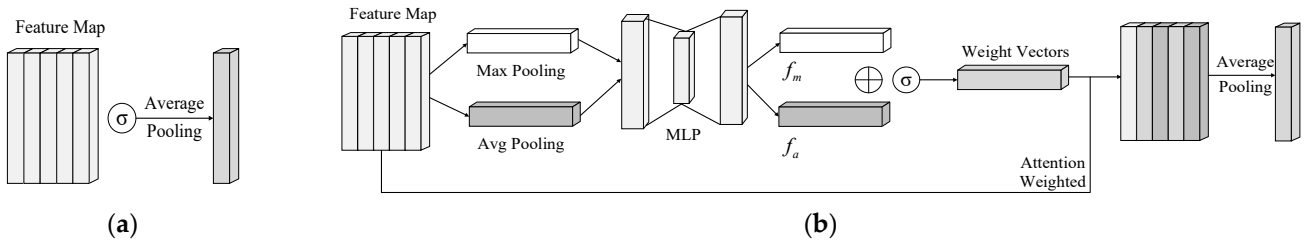

Figure 6. Field feature fusion process. (**a**) Directly concatenated; (**b**) channel attention mechanism.

To address this, unlike Li [26], who used the PCA method, we drew inspiration from the channel attention mechanism in computer vision and implemented a feature fusion method based on attention weighting. This method can express the feature information of the request packet in a high-dimensional manner. The processing flow of this method is shown in Figure 6b below.

Compared with the directly connected field features shown in Figure 6a, the field feature fusion vector obtained through our method can measure the importance of different features, which are represented by different shades of color in Figure 6b. This enables the model to focus better on the attack characteristics of malicious requests during the training process, thereby improving the performance of the detection model.

The input to this module is a feature matrix W composed of the feature vectors of each field:

$$W = [w_1, w_2, \ldots, w_C] \tag{5}$$

The size of the matrix W is $H \times C$, where H is the length of each feature vector and C is the number of feature vectors. This method first applies global max pooling and global average pooling to the matrix, resulting in two descriptor vectors of size $1 \times C$. These vectors are then fed into a two-layer fully connected neural network with shared parameters. The first layer has C/r neurons with the ReLU activation function, and the second layer has C neurons, producing the corresponding output vectors f_a and f_m:

$$f_a = \mathrm{MLP}(\mathrm{AvgPool}(W)) \tag{6}$$

$$f_m = \mathrm{MLP}(\mathrm{MaxPool}(W)) \tag{7}$$

In the formula, $\mathrm{MLP}(\cdot)$ represents the fully connected neural network, $\mathrm{AvgPool}(\cdot)$ is the average pooling function, and $\mathrm{MaxPool}(\cdot)$ is the max pooling operation. The two resulting vectors are added together and transformed using the sigmoid activation function to obtain the weight coefficient vector corresponding to each feature vector M, as shown in Equation (8), where σ denotes the sigmoid activation function.

$$M = \sigma(f_a + f_m) \tag{8}$$

For each feature w_i in the feature matrix W, it is weighted based on the corresponding weight coefficient in $\alpha\alpha$, resulting in the attention-weighted feature matrix $H = [h_1, h_2, \ldots, h_C]$ for the HTTP request, where $h_i = w_i \times m_i$. Finally, by applying an average pooling operation to H, the final field feature fusion vector h_a is obtained, as shown in Equation (9):

$$h_a = \mathrm{AvgPool}(H) \tag{9}$$

5.3. Statistical Feature Selection Based on Random Forest

This paper analyzed statistical attributes and obtained a total of 22 attribute features that can be used to distinguish malicious requests. However, for these statistical features, their feature distributions may be relatively concentrated and there may be correlations between different statistical features, leading to redundancy in the representation of HTTP request statistical features. On the other hand, some statistical features may be irrelevant for certain HTTP request datasets, which means they could provide little or even negative contributions to distinguishing malicious requests.

This paper evaluated the importance of each statistical feature in the request packets based on the Gini coefficient [27]. This metric measures the probability of randomly selecting from the dataset two different samples that belong to different classes. The lower this probability, the higher the purity of the dataset. Specifically, for a given training set D, the Gini coefficient $\mathrm{Gini}(D)$ is defined as follows:

$$\mathrm{Gini}(D) = 1 - \sum_{i=1}^{m} p(x_i)^2 \tag{10}$$

Here, $p(x_i)$ is the probability of class x_i appearing in the set, and m is the number of classes. Furthermore, assuming the dataset is divided using feature A, resulting in m sample subsets D_i, and the proportion of samples belonging to the k-th class in subset D_i is p_k^i, the Gini coefficient of feature A can be defined as follows:

$$\text{Gini}(D, A) = \sum_{i=1}^{m} \frac{|D_i|}{|D|} \text{Gini}(D_i) \tag{11}$$

Therefore, the importance score of a statistical feature can be represented by the reduction in the Gini coefficient before and after the sample division. For feature A, its importance score can be calculated using Equation (12):

$$\text{im}_f(A) = \text{Gini}(D) - \text{Gini}(D, A) \tag{12}$$

If the value of $\text{im}_f(A)$ is large, this indicates that feature A contributes more to the classification task of malicious requests, and thus the importance of feature A is higher.

Furthermore, by setting a threshold, features irrelevant to the malicious request detection task can be eliminated, reducing redundancy among different statistical features to some extent and selecting a relatively optimal subset of statistical features. Our model implements a feature selection method based on importance scores using the Random Forest algorithm. It calculates the importance score of each feature by measuring the difference in the Gini coefficient before and after node classification in the decision trees and selects statistical features by setting a g_threshold.

Specifically, for a given training set D, let its statistical feature set be $F = \{f_1, f_2, \ldots, f_c\}$. The Random Forest algorithm first performs random sampling on the training set D to obtain n training subsets D_i. Then, different decision tree models are trained on each training subset. For the training subset D_i, let its corresponding decision tree model be Tree_i. The importance of statistical feature f_j at the nodes of this decision tree can be represented by the change in the Gini coefficient before and after splitting, as shown in Equation (13):

$$\text{im}_{i,j,m}^g = \text{GI}_m - \text{GI}_l - \text{GI}_r \tag{13}$$

where GI_m represents the Gini coefficient of node m and GI_l and GI_r represent the Gini coefficients of the left and right child nodes after splitting, respectively. For the set of nodes M where the statistical feature f_j appears in the decision tree Tree_i, the importance of feature f_j in the decision tree Tree_i can be calculated with the following:

$$\text{im}_{i,j}^g = \sum_{m \in M} \text{im}_{i,j,m}^g \tag{14}$$

For the n decision tree models in the Random Forest, the importance of feature f_j in the Random Forest can be calculated as:

$$\text{im}_j^g = \sum_{i=1}^{n} \text{im}_{i,j}^g \tag{15}$$

Using the above calculation method, the importance values, $\text{IM} = \left\{\text{im}_1^g, \text{im}_2^g, \ldots, \text{im}_c^g\right\}$, of each feature in the statistical feature set $F = \{f_1, f_2, \ldots, f_c\}$ can be obtained. By normalizing these values, the importance scores of each statistical feature can be derived as follows:

$$\text{im}_j = \frac{\text{im}_j^g}{\sum_{i=1}^{c} \text{im}_i^g} \tag{16}$$

Finally, after obtaining the importance scores of each feature, the g_threshold is set. Each feature is compared against this g_threshold. If the importance of the feature is higher than the g_threshold, the feature is selected to form the statistical feature vector; otherwise, the feature is discarded. We conducted experiments on g_threshold selection in Section 6.3. This process results in the statistical feature vector for the HTTP request. The algorithm flow is shown in Algorithm 1 below. Based on this algorithm, dynamic selection of statistical features for request packets can be achieved, yielding a statistical feature vector that significantly contributes to the detection of malicious requests.

Algorithm 1 Feature Selection Method

Input: Training set D, statistical set F, importance threshold g_threshold
Output: Statistical feature set after feature selection F_{set}
Randomly sample the training set D to obtain n training subsets D_i
for any training subset $D_i \in D$ **do**
    train the decision tree model and generate a decision tree $Tree_i$
    for any feature $f_j \in F$ **do**:
        for any node $m \in Tree_i$ **do**
            if $f_j \in m$ **then**
                calculate the Gini coefficient GI_m of node m
                calculate the Gini coefficient of the left and right child nodes GI_l, GI_r
                make $im^g_{i,j,m} = GI_m - GI_l - GI_r$
            end if
            update $im^g_{i,j} = im^g_{i,j} + im^g_{i,j,m}$
        update $im^g_i += im^g_{i,j}$
        end for
    end for
end for
Normalize $im^g_i \in im^g$
for any feature $f_j \in F$ **do**
    if $im_j \geq$ g_threshold **then**
        Update collection F_{set} add f_j
    end if
end for

Additionally, since the values of different statistical features may have different scales, directly using the vector composed of these features to train the malicious request detection model may cause the model to focus more on features with larger values during training while ignoring features with relatively smaller values. This can distort the learning effect of the model and reduce its training speed. Therefore, to ensure a more balanced contribution of each statistical feature to the model, this paper standardized the obtained statistical features, ensuring that each feature dimension has a mean of 0 and a standard deviation of 1, approximating a standard normal distribution. The standardization function is calculated as shown in Equation (17):

$$x' = \frac{x - \overline{x}}{\sigma} \tag{17}$$

where x is the original feature value of the statistical feature, $\overline{x}$ is the mean of the feature, σ is the standard deviation of the original feature, and x' is the standardized feature value. After standardizing each statistical feature, they can be concatenated to obtain the final statistical feature vector.

5.4. Dual-Branch Classification Network

After obtaining the field fusion feature vector and the statistical feature vector of the request packet, to ensure that these feature vectors contribute more evenly to the training of the detection model, we propose a dual-branch classification network model. Its structure is shown in Figure 7 below, where the gray background represents the model architecture and the white background represents the vector.

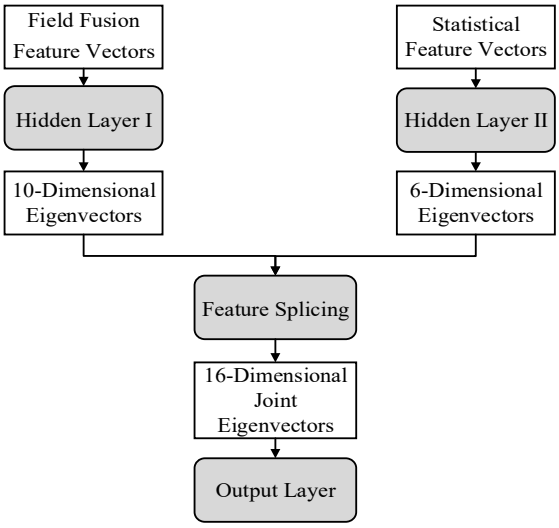

Figure 7. Dual-branch classification network model structure.

This model uses separate hidden layers consisting of a single layer of inactive linear fully connected layers in different branches to transform and represent the field fusion feature vector and the statistical feature vector, resulting in a 10-dimensional feature vector and a 6-dimensional feature vector. These two feature vectors are then concatenated, and the final 16-dimensional combined feature vector is classified through the output layer. We use Softmax as an activation function in the output layer to obtain the final classification result, using cross entropy loss as the loss function.

Compared to a general multilayer perceptron (MLP) classification model, this model first maps the field fusion feature vector and the statistical feature vector through different hidden layers instead of directly concatenating the feature vectors. This allows each hidden layer to learn the optimal parameter weights for the two types of features and mitigates overfitting or underfitting issues when the input feature dimensions are uneven. As a result, the model can more effectively capture the complexity and diversity of the features.

6. Experimental

6.1. Evaluation Metrics

First, regarding the selection of evaluation metrics [28] for classification problems, there are many metrics available, and the appropriate ones should be chosen based on the specific application scenario to comprehensively assess the performance of the algorithm. For malicious request classification results, they can be divided into four types: TP, FP, TN, and FN. Here, T and F represent whether the predicted result matches the actual result (true if they match, false otherwise), while P and N indicate whether the sample is predicted as the fast path or the slow path. Based on these results, four common metrics can be calculated, accuracy, precision, recall, and F1 score, as shown in Equation (18) below.

$$\begin{cases} \text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \\ \text{precesion} = \frac{\text{TP}}{\text{TP} + \text{FP}} \\ \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \\ \text{F}_1 = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \end{cases} \tag{18}$$

The classifier needs to effectively separate normal requests from malicious requests while minimizing the occurrence of false positive (FP) cases. Therefore, in the context of malicious request classification, the most important metrics are accuracy and precision.

6.2. Datasets

6.2.1. Csic 2010

The CSIC 2010 dataset [29] is a publicly available HTTP request dataset released by the Spanish National Research Council (CSIC) in 2010. It contains 36,000 normal requests and approximately 25,000 malicious requests, and is widely used for performance testing of various web attack protection systems. The HTTP request packet data in this dataset is automatically generated based on specific rules by simulating user access behaviors such as registration, product browsing, and purchasing on an e-commerce website. The types of malicious request data include SQL injection, CRLF injection, XSS attacks, file traversal attacks, and more. We divided the positive and negative samples in the CSIC 2010 dataset into training and testing sets in a 4:1 ratio, and further split the validation set from the training set for stratified five-fold cross validation.

6.2.2. Cloud Service Dataset

Although the CSIC dataset has been widely used in the research field of malicious request detection, it is not based on cloud service traffic data and differs significantly from real-world applications. This paper constructed the Cloud Service dataset (Label CND dataset) based on real cloud service logs provided by a cloud service provider.

The cloud service provider uses a Web Application Firewall (WAF) to protect business data traffic of various web applications in different business scenarios, recording massive amounts of HTTP request access logs and quarantine logs. However, due to incomplete protection rules set by some users in practical applications, the WAF frequently misjudges or misses attacks, making it impossible to use the WAF's judgment results as labels for the log data. Instead, expert knowledge is required for analysis and judgment.

The Label CND dataset contains approximately 14,000 normal request logs and 12,000 malicious request logs. The data labels are manually annotated by cybersecurity experts, covering attacks such as CC attacks, SQL injection, and crawler attacks. We adopted the same dataset partitioning strategy and stratified five-fold cross validation on the Label CND dataset.

6.3. Experimental Results and Analysis

6.3.1. Selection of Model Parameters

In the proposed model, the dimensionality of the character embedding vectors f_dim and the threshold for statistical feature importance scores g_threshold significantly impact the detection results of malicious requests. This section conducted multiple experiments to determine the optimal parameter values, thereby improving the accuracy and generalization ability of the detection model.

First, this paper set the dimensionality of character embedding vectors to 15, 30, 45, 60, 75, and 90 while keeping other model parameters unchanged. Experiments were conducted on the HTTP CSIC 2010 dataset and the Label CND dataset, and the accuracy metrics of the model under the corresponding parameters were calculated, as shown in Figure 8a

below. From the experimental results, it can be observed that the value of the f_dim parameter has a significant impact on the model's accuracy metrics. On both the CSIC 2010 and Label CND datasets, the detection accuracy of the malicious request detection model initially increased as the parameter value increased. However, when the value of f_dim reached 60, the model's accuracy metrics tended to stabilize. This indicates that as the value of f_dim increases, the character embedding matrix corresponding to the request packet fields can more accurately reflect the attack characteristics of malicious requests. When the value of f_dim exceeds a certain threshold, the feature information retained by the character embedding matrix reaches a bottleneck, and the model's detection rate for malicious requests stabilizes. Therefore, in both the CSIC 2010 and Label CND datasets, the value of f_dim for the model was set to 60, meaning the dimensionality of the character embedding vectors obtained through the Embedding layer is 60.

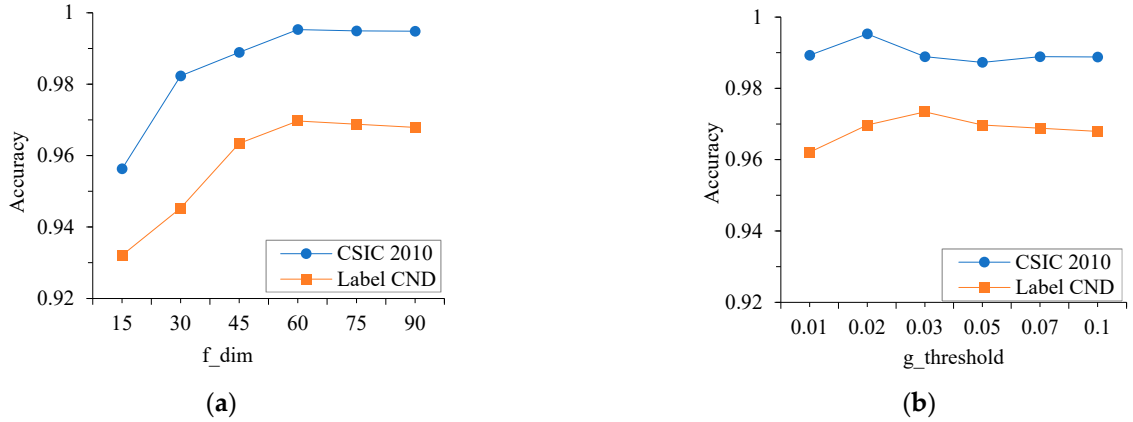

Figure 8. The impact of different parameter values on model performance. (**a**) f_dim parameter; (**b**) g_threshold parameter.

For the statistical features of HTTP requests, our paper first calculated the importance scores of each statistical feature on the CSIC 2010 dataset and the Label CND dataset, as shown in Figure 9 below. On different datasets, due to the composition of the anomaly attack types and the specific forms of the anomaly attacks, varying degrees of feature importance are displayed. For example, the 'param_len' of abnormal attacks on the CSIC2010 dataset is significantly longer than normal samples compared to the CND dataset, making it easy to distinguish samples from the CSIC2010 dataset through the 'param_len' parameter, resulting in high feature importance.

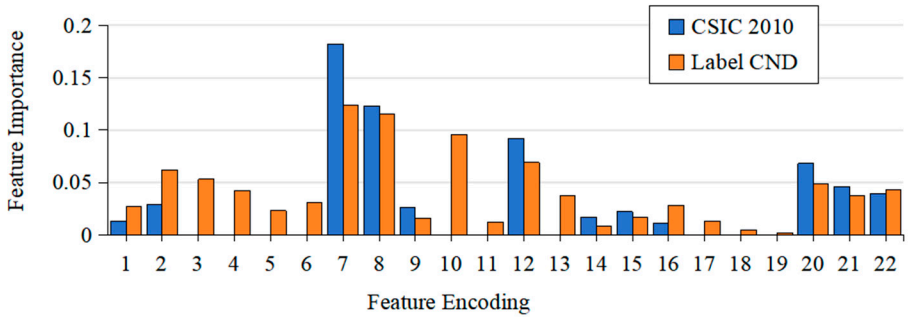

Figure 9. Importance of each statistical feature in different datasets.

This paper set the parameter g_threshold to 0.01, 0.02, 0.03, 0.05, 0.07, and 0.1 as thresholds for statistical feature selection. Experiments were conducted on the CSIC 2010 dataset and the Label CND dataset to investigate the impact of the g_threshold on the model's detection results. The results are shown in Figure 8b above. It can be observed

that the value of the g_threshold significantly affects the model's accuracy. On both the CSIC 2010 and Label CND datasets, as the g_threshold increased, the accuracy of the detection model first rose and then declined. This is because when the g_threshold is too small, more irrelevant features are included as input to the detection model, negatively impacting its performance. Conversely, when the g_threshold is too large, some valuable statistical features are filtered out, leading to an incomplete representation of the attack characteristics of malicious requests. Therefore, in the CSIC 2010 dataset, the g_threshold was set to 0.02, where the model's detection rate reaches its peak, corresponding to the selection of eight statistical features. In the Label CND dataset, the g_threshold was set to 0.03, where the model achieved the highest accuracy, corresponding to the selection of 13 statistical features.

6.3.2. Effectiveness of Methods

To verify the effectiveness of the field feature fusion method proposed in this paper, under the condition that other model parameters remain unchanged, the field feature vectors of the request packets were processed using both feature concatenation and attention-based weighted fusion methods to obtain the field fusion feature vectors. The model was then trained on the CSIC 2010 and Label CND datasets.

From the experimental results shown in Figure 10, it can be observed that on the CSIC 2010 dataset, all detection metrics of the model exceeded 99%. On the Label CND dataset, the accuracy metric of the proposed detection model reached 96.97%, and the values of the other metrics also exceeded 96%, demonstrating the outstanding detection capability of the proposed method for malicious requests. Additionally, on both the CSIC 2010 and Label CND datasets, all metrics of the proposed detection method were higher than those of the model using feature concatenation, proving the effectiveness of the field feature fusion method. Unlike when directly concatenating field features, the field feature fusion vector obtained through attention-based weighted fusion can measure the importance of different features, enabling the model to better focus on the attack characteristics of malicious requests during training, thereby improving the performance of the detection model.

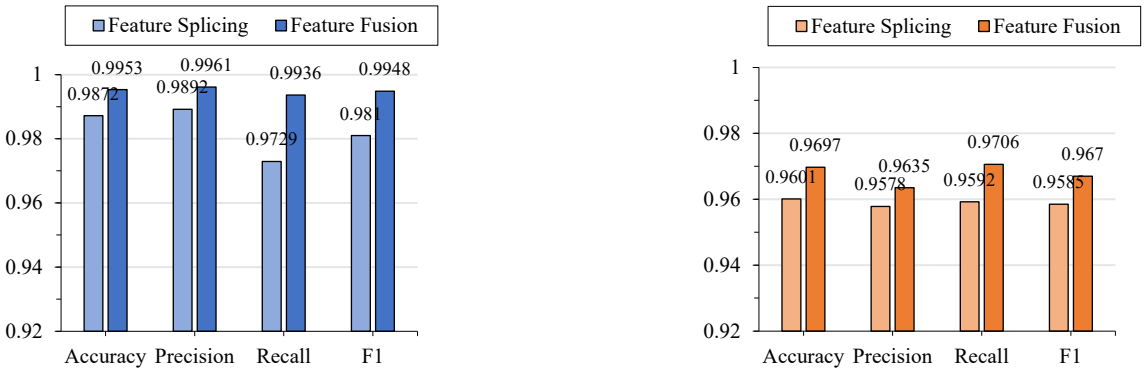

Figure 10. Performance analysis of field feature fusion methods on different datasets. (**a**) HTTP CSIC 2010 dataset; (**b**) Label CND dataset.

Furthermore, to verify the effectiveness of the statistical feature selection method proposed in this paper, the model was trained without statistical feature selection while keeping the other model parameters unchanged. In this case, all statistical features were used as input to the model, and experiments were conducted on the CSIC 2010 and Label CND datasets. As shown in Figure 11, on the CSIC 2010 dataset, the proposed statistical feature selection method significantly improved the model's detection performance. The

accuracy, precision, recall, and F1 score increased by approximately 2.4%, 1.4%, 2.8%, and 2.1%, respectively, compared to the detection method without feature selection. This indicates that the proposed statistical feature selection method can obtain the statistical features most relevant to attack characteristics on the CSIC 2010 dataset, including the length of transmission parameters, the number of parameters, the length of the longest parameter, etc., while eliminating irrelevant features that contribute little or even negatively to improvements in the model performance, thereby improving the model's detection rate for malicious requests. Similarly, on the Label CND dataset, the proposed statistical feature selection method can also remove irrelevant statistical features and eliminate redundancy among different features, enabling the extracted statistical feature vector to more effectively reflect the data characteristics of the request packets. This effectively enhances the detection performance of the proposed model for attacks.

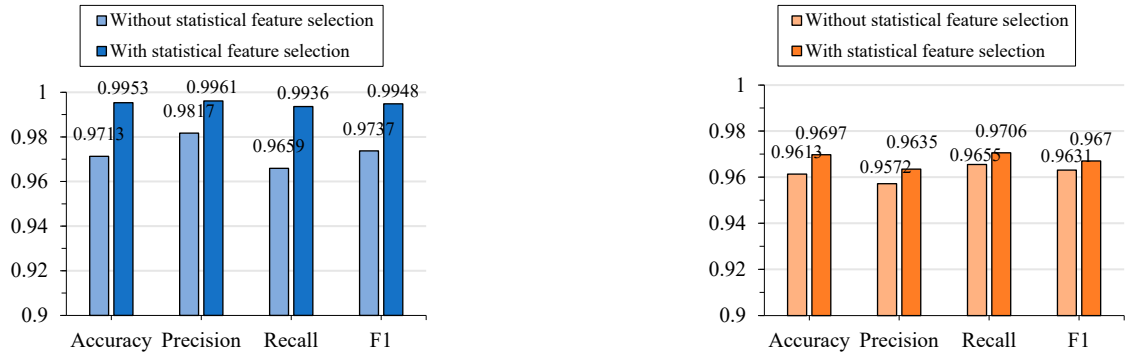

Figure 11. Performance analysis of statistical feature selection methods on different datasets. (**a**) HTTP CSIC 2010 dataset; (**b**) Label CND dataset.

6.3.3. Comparison with Baseline Model

Finally, to further validate the detection effectiveness of the proposed method, this paper selected the research methods from refs. [15,30–32] for comparative experiments on the CSIC 2010 and Label CND datasets. The experimental results are shown in Figure 12 below. Among them, SVM [30] and TF-IDF [15] are traditional methods, while CNN-GRU [31] and Bert [32] are relatively novel methods.

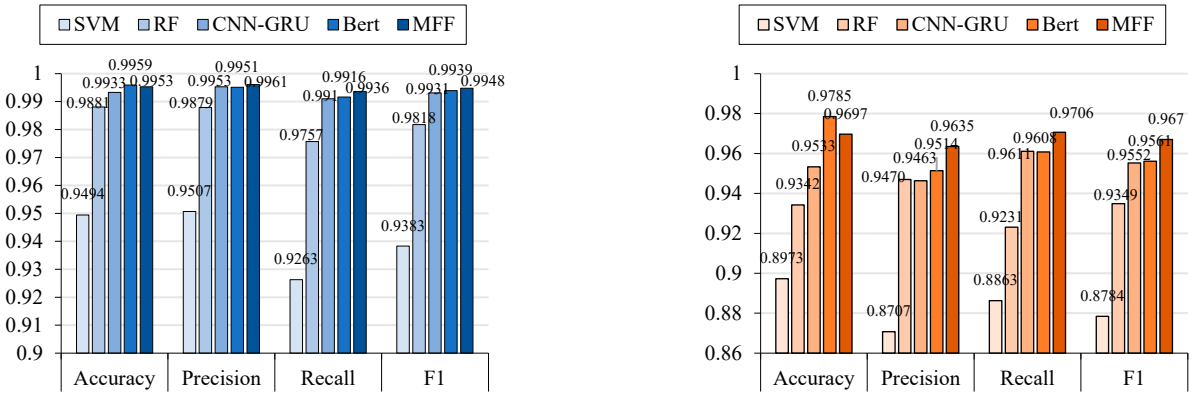

Figure 12. Experimental comparison of this method with other methods on different datasets. (**a**) HTTP CSIC 2010 dataset; (**b**) Label CND dataset.

Among them, ref. [30] used the maximum relevance–minimum redundancy method to select statistical features from request packets and then implemented malicious request detection and classification based on an SVM model. However, since it only extracts statistical features from the request packets and does not consider the semantic information of the field texts during detection and classification, this method performs poorly on complex datasets (e.g., the Label CND dataset). Its accuracy, precision, recall, and F1 score show a noticeable gap compared to other detection methods.

Ref. [15] extracts URL field features from request packets, converts the URL field features into word vectors using the n-gram and TF-IDF methods, and then classifies malicious requests using the Random Forest (RF) algorithm. This method achieved relatively good detection performance on the CSIC 2010 dataset. However, since the malicious request data in the Label CND dataset is more complex and diverse, its attack characteristics cannot be fully represented by the URL field alone. As a result, the accuracy and recall of this method on the Label CND dataset were only 93.42% and 92.31%, indicating a potential for higher false positives and false negatives.

Ref. [31] combined a CNN network with a GRU network to implement a CNN-GRU detection model, which extracts both field features and statistical features of malicious requests to detect and identify malicious requests. This method comprehensively considers the field attributes and statistical attributes of request packets, achieving better detection performance on both the CSIC 2010 and Label CND datasets compared to the previous two methods. However, this method does not perform feature fusion on the field features of the request packets but directly concatenates the fields before inputting them into the CNN-GRU network for feature extraction. In addition, the statistical feature set selected for the request packets is static and fixed, without feature analysis and selection tailored to specific datasets. Therefore, there is still room for optimization of this method on the Label CND dataset.

Ref. [32] is based on the Bert model for word vector representation learning of request messages and it integrated the Transformer network for feature extraction and classification. Compared with traditional methods, this type of method has higher feature extraction capabilities and can better extract semantic information from malicious requests. Therefore, in the experimental results, the classification performance of the model was relatively good. This type of method is currently a hot topic in malicious request detection research, but considering the difficulty in obtaining the true labels of malicious requests in real networks, insufficient training data may affect the detection performance of such models.

7. Conclusions

Application-layer injection attacks are a hot topic in the field of cloud service security. Rule-based filtering methods cannot identify unknown attacks, and traditional machine learning methods suffer from single-dimensional feature extraction, failing to fully utilize the rich information in cloud service logs, resulting in weak model generalization capabilities. Therefore, we propose a malicious request detection method based on multi-feature fusion. Feature extraction is performed from two dimensions: the field attributes and statistical attributes of request packets. For field attributes, the TextCNN network is used to extract field features, followed by field feature fusion based on an attention mechanism. For statistical attributes, the Gini coefficient is used to evaluate feature importance, and the random forest algorithm is employed for feature selection and vector construction. Additionally, a multi-layer network classification model with a branch structure is designed to jointly train field feature vectors and statistical feature vectors. Extensive comparative experiments on the CISC 2010 dataset and the Label CND dataset demonstrated that the

proposed malicious request detection model outperforms existing methods in terms of detection performance.

Author Contributions: Conceptualization, Z.C.; methodology, Z.C. and C.D.; software, C.D.; validation, X.G.; formal analysis, C.D.; investigation, X.L.; resources, Z.C.; data curation, C.D.; writing—original draft preparation, Z.C. and C.D.; writing—review and editing, H.H. and X.L.; visualization, C.D.; supervision, H.H.; project administration, Z.C.; funding acquisition, H.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Fundamental Research Funds for the Central Universities under grant number ZYGX2022T001 and Sichuan Province Science and Technology Project under grant number 2022ZHCG0133.

Data Availability Statement: The datasets presented in this article are not readily available because the data are part of an ongoing study. Requests to access the datasets should be directed to dengchen@std.uestc.edu.cn.

Conflicts of Interest: Author Zhouguo Chen and Xinze Li were employed by the company: The 30th Research Institute of China Electronics Technology Group Corporation. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BERT	Bidirectional Encoder Representations from Transformers
DistilBERT	Distilled Bidirectional Encoder Representations from Transformers
FedGAT	Federated Graph Attention Network
FWaaS	Firewall as a Service
LSTM	Long Short-Term Memory
MFF	Multiple Feature Fusion
PaaS	Platform as a Service
RF	Random Forest
RTIDS	Robust Transformer-based Intrusion Detection System
SaaS	Software as a Service
TextCNN	Convolutional Neural Networks for Sentence Classification
WAF	Web Application Firewalls

References

1. Cybersecurity Insiders. Cloud Security Report. 2024. Available online: http://webobjects2.cdw.com/is/content/CDW/cdw/on-domain-cdw/brands/check-point/2024-cloud-security-report-checkpoint-final-1cleaned.pdf (accessed on 16 January 2025).
2. Qu, Z.; Ling, X.; Wang, T.; Chen, X.; Ji, S.; Wu, C. AdvSQLi: Generating Adversarial SQL Injections Against Real-World WAF-as-a-Service. *IEEE Trans. Inf. Forensics Secur.* **2024**, *19*, 2623–2638. [CrossRef]
3. CYBRARY. 2024's Data Breaches, Incidents, and How to Avoid Them in 2025. 2025. Available online: https://www.cybrary.it/blog/2024-data-breaches-how-avoid-them-2025 (accessed on 16 January 2025).
4. Li, J.; Jiang, H.; Jiang, W.; Wu, J.; Du, W. SDN-based Stateful Firewall for Cloud. In Proceedings of the 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), Baltimore, MD, USA, 25–27 May 2020; pp. 157–161.
5. Pavithra, B.; C, V.; Mishra, N.; Naveen, G. Cloud Security Analysis using Machine Learning Algorithms. In Proceedings of the 2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS), Trichy, India, 23–25 August 2023; pp. 704–708.
6. Sanagana, D.P.R.; Tummalachervu, C.K. Securing Cloud Computing Environment via Optimal Deep Learning-based Intrusion Detection Systems. In Proceedings of the 2024 Second International Conference on Data Science and Information System (ICDSIS), Hassan, India, 17–18 May 2024; pp. 1–6.

7. Chethan, M.S.; Channakrishnaraju; Rajeswari, R.; Selvam, M. Cyber Attack Detection System in University Private Cloud Using Machine Learning. In Proceedings of the 2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS), Erode, India, 18–20 October 2023; pp. 1080–1085.
8. Rajapraveen, K.N.; Pasumarty, R. A Machine Learning Approach for DDoS Prevention System in Cloud Computing Environment. In Proceedings of the 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bangalore, India, 16–18 December 2021; pp. 1–6.
9. Kwedza, P.; Chindipha, S.D. Cryptojacking Detection in Cloud Infrastructure Using Network Traffic. In Proceedings of the 2023 International Conference on Electrical, Computer and Energy Technologies (ICECET), Changchun, China, 22–24 September 2023; pp. 1–6.
10. Denning, D.E. An Intrusion-Detection Model. *IEEE Trans. Softw. Eng.* **1987**, *SE-13*, 222–232. [CrossRef]
11. Eslahi, M.; Hashim, H.; Tahir, N.M. An efficient false alarm reduction approach in HTTP-based botnet detection. In Proceedings of the 2013 IEEE Symposium on Computers & Informatics (ISCI), Langkawi, Malaysia, 7–9 April 2013; pp. 201–205.
12. Krügel, C.; Toth, T.; Kirda, E. Service specific anomaly detection for network intrusion detection. In Proceedings of the 2002 ACM Symposium on Applied Computing, Madrid, Spain, 11–14 March 2002; pp. 201–208.
13. Setiyaji, A.; Ramli, K.; Hidayatulloh, Z.Y.; Budhi Dharmawan, G.S. A technique utilizing Machine Learning and Convolutional Neural Networks (CNN) for the identification of SQL Injection Attacks. In Proceedings of the 2024 4th International Conference of Science and Information Technology in Smart Administration (ICSINTESA), Balikpapan, Indonesia, 12–13 July 2024; pp. 1–6.
14. Zhang, X.; Meng, F.; Xu, J. PerfInsight: A Robust Clustering-Based Abnormal Behavior Detection System for Large-Scale Cloud. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2–7 July 2018; pp. 896–899.
15. Ramezany, S.; Setthawong, R.; Tanprasert, T. A Machine Learning-based Malicious Payload Detection and Classification Framework for New Web Attacks. In Proceedings of the 2022 19th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Prachuap Khiri Khan, Thailand, 24–27 May 2022; pp. 1–4.
16. Ma, J.; Saul, L.K.; Savage, S.; Voelker, G.M. Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 1245–1254.
17. Kim, H.; Yoon, Y. An Ensemble of Text Convolutional Neural Networks and Multi-Head Attention Layers for Classifying Threats in Network Packets. *Electronics* **2023**, *12*, 4253. [CrossRef]
18. Xie, X.; Ren, C.; Fu, Y.; Xu, J.; Guo, J. SQL Injection Detection for Web Applications Based on Elastic-Pooling CNN. *IEEE Access* **2019**, *7*, 151475–151481. [CrossRef]
19. Wu, Z.; Zhang, H.; Wang, P.; Sun, Z. RTIDS: A Robust Transformer-Based Approach for Intrusion Detection System. *IEEE Access* **2022**, *10*, 64375–64387. [CrossRef]
20. Momu, S.A.; Siddiqui, R.R.; Hoque, M.S.; Kawsur, M.M. Evaluating the Performance of Long Short-Term Memory for Web Attack Detection. In Proceedings of the 2023 26th International Conference on Computer and Information Technology (ICCIT), Cox's Bazar, Bangladesh, 13–15 December 2023; pp. 1–6.
21. Bokolo, B.G.; Chen, L.; Liu, Q. Detection of Web-Attack using DistilBERT, RNN, and LSTM. In Proceedings of the 2023 11th International Symposium on Digital Forensics and Security (ISDFS), Chattanooga, TN, USA, 11–12 May 2023; pp. 1–6.
22. Jianping, W.; Guangqiu, Q.; Chunming, W.; Weiwei, J.; Jiahe, J. Federated learning for network attack detection using attention-based graph neural networks. *Sci. Rep.* **2024**, *14*, 19088. [CrossRef] [PubMed]
23. Shannon, C.E. A Mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
24. Yu, X.; Meng, W.; Liu, Y.; Zhou, F. TridentShell: An enhanced covert and Scalable Backdoor Injection Attack on Web Applications. *J. Netw. Comput. Appl.* **2024**, *223*, 103823. [CrossRef]
25. Liu, Z. Research on SQL Injection Detection Based on Deep Learning. In Proceedings of the 2023 IEEE 7th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 15–17 September 2023; pp. 746–749.
26. Li, M.; Han, D.; Li, D.; Liu, H.; Chang, C.-C. MFVT: An anomaly traffic detection method merging feature fusion network and vision transformer architecture. *EURASIP J. Wirel. Commun. Netw.* **2022**, *2022*, 39. [CrossRef]
27. Disha, R.A.; Waheed, S. Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique. *Cybersecurity* **2022**, *5*, 1. [CrossRef]
28. Liu, X.; Liu, J. Malicious traffic detection combined deep neural network with hierarchical attention mechanism. *Sci. Rep.* **2021**, *11*, 12363. [CrossRef] [PubMed]
29. Nguyen, H.T.; Torrano-Gimenez, C.; Álvarez, G.; Petrovic, S.; Franke, K. Application of the Generic Feature Selection Measure in Detection of Web Attacks. In Proceedings of the Computational Intelligence in Security for Information Systems 4th International Conference, Torremolinos-Málaga, Spain, 8–10 June 2011; pp. 25–34.

30. Rajagopal Smitha, K.S.H.; Poornima Panduranga Kundapur, A. Machine Learning Approach for Web Intrusion Detection: MAMLS Perspective. In Proceedings of the Soft Computing and Signal Processing, Hyderabad, India, 21–22 June 2019.
31. Niu, Q.; Li, X. A High-performance Web Attack Detection Method based on CNN-GRU Model. In Proceedings of the 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 12–14 June 2020; pp. 804–808.
32. Fan, Y.; Liu, X.; Zhu, J.; Jiang, Q.; Li, Q.; Xu, D. Research on Web Attack Detection of Service Websites Based on BERT. *Comput. Technol. Dev.* **2022**, *32*, 168–173. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.