INTERNATIONAL JOURNAL OF ACADEMIC RESEARCH IN BUSINESS & SOCIAL SCIENCES

Vol 15, Issue 11, (2025) E-ISSN: 2222-6990

Research on Feature Modeling and Preprocessing Methods for Cross-Site Scripting Attack Detection

Huang Li, Dr. Amirrudin Kamsin

Faculty of Information Technology, City University Malaysia, Kuala Lumpur, Malaysia

Corresponding Author Email: 6595580@qq.com

DOI Link: http://dx.doi.org/10.6007/IJARBSS/v15-i11/26948

Published Date: 14 November 2025

Abstract

Cross site scripting attacks (XSS) are one of the main security threats facing current web applications, and attackers often use multi-layer encoding and obfuscation techniques to evade traditional detection mechanisms. This paper proposes an improved feature modeling and preprocessing method to address the issues of incomplete semantic restoration of multiple obfuscated scripts in the preprocessing stage and ignoring cross domain contextual associations between HTML and JavaScript in the feature modeling stage. The core of this method includes: 1) a layered trigger decoding mechanism that achieves efficient and accurate semantic restoration of complex obfuscated payloads through a detection driven recursive decoding strategy; 2) A domain aware feature modeling method that combines HTML structural features with JavaScript behavioral features to construct a joint feature representation that captures cross domain semantics. To verify the effectiveness of the method, this paper crawled and annotated a dataset containing 30000 samples based on the open-source XSS vulnerability library XSSed and real network traffic. The experimental results on this dataset show that the proposed method outperforms traditional preprocessing processes in terms of accuracy, recall, and F1 score. In addition, compared with mainstream deep learning models, this method improves inference speed by only 2.1 ms/sample while maintaining a considerable detection accuracy of F1 Score 99.59%, demonstrating its practical value and potential in web security applications that require high real-time performance and low resource consumption.

Keywords: XSS Detection, Feature Modeling, Data Preprocessing, Confusion Restoration, Domain Aware Segmentation, Lightweight Testing

Introduction

With the continuous evolution of Web technology and the rapid growth of Internet applications, browser based security threats show a complex and diversified development trend. Cross Site Scripting (XSS) is one of the most common and widely affected vulnerability types. According to the statistical results of the Open Web Application Security Project

(OWASP, 2021), XSS attacks have long been at the forefront of Web application security risks. By injecting malicious script into the web page or request, the attacker makes the browser execute illegal instructions without knowing it, so as to achieve session hijacking, sensitive information theft, page tampering and other malicious behaviors (Alsaffar et al., 2022). In recent years, with the development of front-end frameworks and dynamic scripting technology, attack tactics have been evolving, and mutation strategies such as code confusion, character splicing, tag deformation, and event attribute embedding have emerged in endlessly, making the traditional detection mechanism face serious challenges (Alaoui&Nfaoui, 2023).

In existing research, traditional XSS detection mainly relies on regular matching, black and white list rules or Web Application Firewall (WAF) policies. This kind of method has certain effect in the face of single-layer coding or fixed pattern attacks, but for attack samples with multi-layer confusion, code nesting and semantic variation, the recognition ability is significantly reduced. In order to overcome this shortcoming, researchers proposed a detection method based on static analysis, dynamic analysis and their combination, which can identify potential risks by parsing the page syntax tree, monitoring runtime behavior, or analyzing data flow dependencies (Gupta, 2016). In recent years, machine learning and deep learning technologies have been widely used in XSS attack detection tasks. By automatically learning the feature distribution of attack samples, a high recognition rate has been achieved (Hamzah et al., 2024). However, on the whole, there are still two main problems in the existing research:

(1) In the pre-processing stage, the existing methods have limited ability to restore the semantics of multiple encoding and deep obfuscation scripts, which often lead to false decoding or information loss, resulting in incomplete feature extraction;
(2) In the feature modeling phase, most methods only model based on word frequency, n-gram or embedded representation, lacking comprehensive modeling of HTML and JavaScript syntax structure and behavior logic, and it is difficult to capture cross domain associated features, thus affecting the generalization performance of detection.

To solve the above problems, this paper proposes a feature modeling and pre-processing method for cross site script attack detection. By improving the data pre-processing process and feature representation mechanism, the model's ability to identify complex confusion and semantic variation loads is improved, so as to improve the detection accuracy, reduce the false positive/false negative rate, and take into account the requirements of lightweight deployment and real-time detection.

The main contributions of this paper are as follows: ① A layered triggered data pre-processing mechanism is proposed to achieve efficient semantic restoration of multiple encoding and deep obfuscation scripts through the detection driven multi-layer decoding process; ② Build a domain aware feature modeling method, integrate HTML tags, event attributes, JavaScript function calls and other domain features, and form a feature expression that emphasizes both structure and behavior; ③ In order to verify the effectiveness of the method, we constructed a dataset that combines the public vulnerability database and real traffic. The experimental results on this basis show that the method is superior to the existing scheme in terms of recognition rate, robustness and generalization ability.

The structure of this paper is as follows: Section 2 reviews the relevant research work, including the preprocessing and obfuscation restoration of XSS attacks, feature modeling and vectorization; Section 3 introduces the overall framework and key algorithms proposed in this paper; Section 4 gives the experimental design, results and analysis; Section 5 summarizes the full text and proposes future research directions.

Related Work

Pretreatment and Confusion Reduction

In XSS attack samples, attackers usually hide malicious scripts in various ways, such as using multi-layer encoding (URL, HTML entity Base64）、 String concatenation, character replacement, event attribute injection, tag confusion, etc. These technologies can effectively circumvent the traditional detection methods based on string matching or regular rules, thus making the attack payload deceptive at the semantic level (Lamrani Alaoui&Nfaoui, 2023).

Early research mainly used basic decoding and cleaning strategies, such as removing comments, unifying character case, performing single-layer URL decoding or HTML entity restoration, to eliminate some confusion. However, this kind of method is limited in the face of multi-layer nesting and mutation coding. Pazos et al. (2020) proposed a detection method based on client template analysis, which restores the execution context of the injected script by parsing the HTML template context. Although the accuracy is improved, this method has high computational complexity and is not suitable for batch data processing. Wang et al. (2010) studied XSS syntax features from the perspective of attack vector structure generation, systematically analyzed attack patterns, and laid a foundation for structural modeling of confusion samples.

In recent years, researchers have begun to introduce mode detection and multi-layer decoding mechanisms in the preprocessing phase to systematically restore complex loads. As an example, some researchers combine regular expressions with context-based analysis to locate nested encoding patterns. Depending on what is found, separate decoding modules are triggered step by step to rebuild the intended semantics (Alsaffar et al., 2022). This detection-oriented preprocessing process generally performs better than static decoding, offering a balance between semantic completeness and computational efficiency.

The existing research shows that high-quality pre-processing mechanism is the premise of successful XSS detection. If the input samples are not decoded correctly, even if the subsequent feature modeling and classification algorithms are more complex, their performance will be significantly limited. Therefore, how to design a "triggering, multi-level, context aware" preprocessing mechanism is still an important research direction in the current XSS attack detection field.

Feature Modeling and Vectorization Methods

After semantic restoration, how to effectively model features of Web script samples is a key link to improve detection performance. Early research mainly used traditional methods based on statistical features, such as word frequency (TF), word frequency inverse document frequency (TF-IDF), word or character n-gram, to map script content into feature representation in vector space. Rathore et al. (2017) combined URL and page content features in a social network environment, and used machine learning classifiers to achieve high recognition accuracy. Such methods have the advantages of simple implementation and

strong interpretability, but they are prone to feature fragmentation and information loss in the face of complex semantic relationships or structural variations.

The development of feature modeling has roughly gone through three stages: (1) from word frequency statistics to n-gram's shallow grammatical features;(2) The deep semantic features from word embedding to sequence model; (3) It is evolving towards structural, semantic and cross domain integration. With the development of deep learning, researchers began to explore methods based on representation learning. Lamrani Alaoui and Nfaoui (2023) proposed a detection model based on the LSTM Encoder Decoder structure, capturing context features through word embedding, and achieving significant performance improvement on multiple public data sets. Li et al. (2022) further combined the CNN–Bi-LSTM–Attention architecture to extract multi-level semantic features, and improved word segmentation algorithm to improve feature consistency. These methods can learn complex semantic patterns in high-dimensional space, but usually ignore the differences between HTML and JavaScript in syntax and behavior, which makes the model have limited generalization ability when dealing with cross domain attack samples.

To solve this problem, the research trend in recent years began to turn to multi domain feature modeling combining structure and semantics. Meng et al. (2022) proposed the joint modeling of DOM structure, event attributes and script functions as feature elements, and tried to describe the "structure behavior" relationship of scripts in the feature space. Zhang et al. (2021) introduced context syntax constraints on the basis of character level modeling to enable the model to recognize attack samples with different coding forms but semantically equivalent. This kind of research shows that in XSS detection tasks, simply relying on statistics or deep embedding is not enough to capture key semantic features, and cross domain structural feature modeling is a key direction to improve detection accuracy and robustness. Existing research has made significant progress in XSS attack detection, but there are still the following shortcomings:

(1) In the preprocessing phase, the lack of a unified, multi-level framework for confusion identification and decoding leads to the inability to effectively recover some attack semantics; (2) In the feature modeling phase, the lack of comprehensive modeling of HTML and JavaScript dual domain syntax and behavior features leads to the lack of generalization ability of the model in diverse attack scenarios.
Based on the above problems, this paper proposes a detection framework with the layered trigger pre-processing mechanism and domain aware feature modeling method as the core, which improves the accuracy and stability of XSS attack detection by enhancing the ability of sample semantic restoration and cross domain feature fusion.

Design of Detection Scheme

Overall Architecture of the Scheme

The challenge in XSS attack detection lies in the fact that attack payloads are frequently subjected to multi-layer encoding and obfuscation, leading to semantic distortion and consequently impairing the effectiveness of feature extraction and classification. To address this issue, this paper introduces improved methods focusing on data preprocessing and feature modeling, thereby constructing a systematic framework tailored for XSS detection. The overall workflow is illustrated in Figure 1.

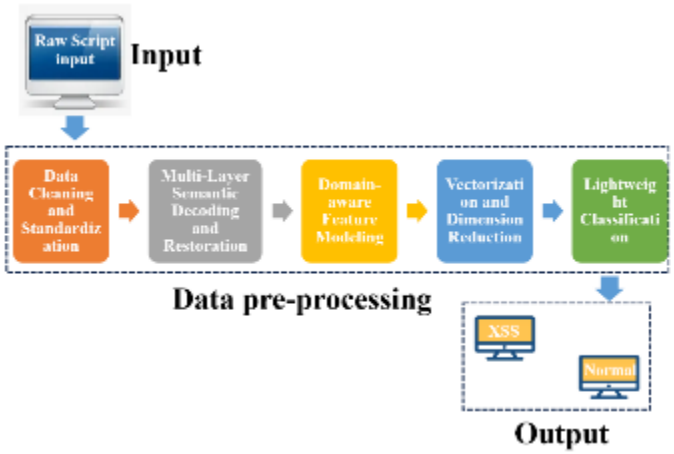

Fig.1 Five-Stage Lightweight XSS Detection Data Preprocessing System

The program includes five main stages:
Stage 1: Data cleaning and normalization: remove invalid samples and exception codes, unify text format, and provide consistent input for subsequent semantic analysis.
Stage 2: Multilayer semantic decoding and restoration: propose a layered triggered decoding mechanism (HDM), activate Base64, URL, HTML entities and other decoding modules as required according to the confusion detection results, and recursively restore the hidden script semantics.
Stage 3: Domain aware word segmentation and feature extraction: Design a domain aware tokenizer to extract the tag, attribute, function call, character n-gram and other features in the HTML structure domain and JavaScript behavior domain, and retain the structure and behavior information of the attack payload.
Stage 4: Feature vectorization and dimension reduction: calculate word level and character level weights based on TF-IDF, and use Truncated SVD to reduce dimensions to generate dense semantic vectors.
Stage 5: Model training and detection decision: use logical regression and support vector machine (SVM) models for classification to evaluate the detection performance of the proposed method under different data sets.
The scheme proposed in this paper uses the process of "hierarchical decoding domain aware feature vector modeling" to realize the closed-loop design from confused samples to structured feature representation, which provides technical support for subsequent experiments and performance analysis.

Layered Trigger Decoding Mechanism
In cross site scripting (XSS) attack samples, attackers usually confuse malicious scripts by means of multi-layer coding, string splicing, event attribute injection or character replacement to avoid static matching or feature detection. If full decoding is performed directly, it will not only increase the computational burden, but also may cause semantic misjudgment. Therefore, this paper proposes a Hierarchical Decoding Mechanism (HDM) to implement multi-layer semantic restoration in a detection driven manner. Efficient and controllable semantic restoration is achieved through "on-demand triggering and recursive compensation", providing standardized input for subsequent feature modeling and lightweight classification.

The overall process of HDM includes three main steps:
Step 1: Confusion detection. Identify potential confusion features through regular matching and context analysis, such as String. fromCharCode(), javascript: protocol or multiple escape characters.
Step 2: Targeted decoding. For the detection results, select the corresponding decoding module for semantic restoration, and update the sample results only when the output contains a valid script structure.
Step 3: Recursive verification and normalization. After each round of decoding, residual confusion is re detected and the format is unified until no new restoration process is triggered. The final output text and processing meta information provide traceability input for subsequent feature extraction.

When the system detects confusion or coding mode, it will automatically call the corresponding decoding module; If no confusion feature is detected, skip this layer processing. After each round of decoding, the confusion detection is performed again. If a new mode is found, it will recursively enter the next layer until the semantics are restored or the set depth is reached. Through this layered trigger mechanism, the system can control the processing complexity, and at the same time, realize the efficient restoration of multi-level confusion samples, so as to improve the semantic consistency and feature integrity of input data, and provide a reliable basis for subsequent domain aware feature modeling.

Confusion Detection and Targeted Decoding
In input samples, attack obfuscation is often reflected in specific syntax and symbol distribution characteristics. In order to achieve lightweight detection, this study uses a recognition strategy based on the fusion of regular expressions and context features. The detection module focuses on identifying the following five types of modes:
(1) String concatenation and construction, such as String. fromCharCode() or suspicious strings generated by plus concatenation;
(2) Code hiding mode, including Base64, URL encoding and multi-level nested escape;
(3) Event attribute calls, such as script logic embedded in attribute values such as onerror, onload, onclick, etc;
(4) Script protocol prefix, such as javascript: or data: and other high-risk protocol headers;
(5) Non standard tags and attributes, including case mixing, missing quotation marks, attribute disorder and other abnormal structures.

The set of confusion tags detected and output determines the subsequent decoding process. If the set is empty, the system will directly output the normalization result; If it is not empty, it will enter the target decoding stage. For different types of confusion, the system calls corresponding lightweight modules, such as Base64 decoding, URL decoding, HTML entity decoding, Unicode/escape restore, or string construction and parsing. All modules follow the standard process of "detection conversion verification". Only when the output results pass the semantic verification (that is, contain valid script structure or logical mode), can they be updated to new sample representation to avoid misjudgment and over decoding.

Recursive Trigger and Result Normalization
In complex multi-layer confusion scenarios, a single decoding often fails to fully restore the original semantics. For this reason, HDM has designed a recursive trigger mechanism to re

execute confusion detection after each decoding round. If a new mode is found, it will automatically enter the next decoding round. This process is constrained by the maximum depth and time threshold to ensure optimal decoding within limited resources.

After the decoding is completed, the system carries out unified and standardized processing on the results, including unified character encoding (UTF-8), HTML tags and attributes lowercase, redundant spaces and control characters removal and other operations.

Domain Awareness Feature Modeling

After being processed by the layered trigger decoding mechanism, the input samples have been transformed from the confused state to the text with clear structure and semantic dominance. After data normalization, the research data has a unified format and consistent semantic expression. However, text samples still exist in the form of natural language or script strings, and do not have the conditions for direct input to machine learning models or neural networks. For this reason, we need to further carry out tokenization and vectorization operations to convert text data into numerical feature representation suitable for computing processing.

The core task of word segmentation is to divide the normalized XSS sample into semantically meaningful smallest units, such as HTML tags, attribute names, JavaScript functions, and symbols, so as to retain the syntax and logic features of the attack payload to the maximum extent at the feature level. Vectorization maps these discrete tokens to a continuous numerical space, enabling the model to learn and reason based on distance calculation, weight distribution and statistical characteristics. Different from general natural language processing tasks, XSS attack detection not only relies on conventional word level statistical patterns, but also must pay attention to character level variation and confusion patterns to ensure that the model can identify confused but semantically equivalent attack samples.

In terms of method selection, this study uses domain aware word segmentation combined with TF-IDF representation of word n-gram and character level n-gram, and further uses Truncated SVD to reduce dimensions to generate low dimensional dense vectors. On the one hand, this method can fully extract the unique structure and confusion features of XSS attacks; On the other hand, the dense representation after dimensionality reduction not only meets the requirements of lightweight neural network (LXDNN) for computing efficiency and memory occupation, but also provides suitable numerical input for the subsequent category balance strategy.

Domain aware Word Segmentation

In the cross site scripting (XSS) detection task, the attack payload is often mixed with HTML tags, event attributes, JavaScript function calls, special symbols, and a variety of obfuscation variants. The traditional natural language processing segmentation strategy based on spaces or punctuation has obvious shortcomings in this scenario, which easily leads to the fragmentation of key attack modes and information loss. Therefore, this research designed and implemented a domain aware tokenizer, which combined with the syntax features in the field of Web security, transformed the normalized text request $d \in D$ into an ordered token sequence for subsequent feature vectorization and model training.

Set the original training data set as:

$$\mathcal{D} = \{d_1, \ldots, d_N\}$$

Where d_i represents the normalized text request. Define word segmentation process as mapping function ϕ:

$$\phi: \mathcal{D} \to \{T^{(d)} | d \in D\}, T^{(d)} = (t_1^{(d)}, t_2^{(d)}, \ldots, t_{m_d}^{(d)})$$

Where $T^{(d)}$ is the token sequence of text d, and m_d is the number of its tokens.

Map a normalized text $d \in \mathcal{D}$ into a series of tokens. The token set consists of two types: word-level tokens and character-level n-grams. Word-level tokens include HTML tag boundaries, attribute names, function names, and so on, with a vocabulary size of V_ω. Character-level n-grams extract sliding substrings of length $n \in \{3,4,5\}$ from the string d, with a vocabulary size of V_c.

Assume a normalized HTTP request payload is given as:

<img src="x" onerror="alert(1)">

After the sample $d \in D$ is processed by the domain aware word splitter $\varphi(\cdot)$, the word segmentation process is shown in Figure 2, which shows the specific word segmentation steps of the sample using the domain aware analyzer.

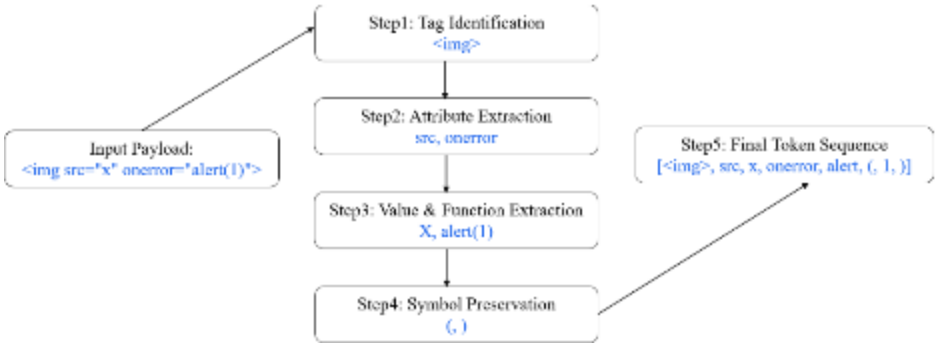

Figure 2 Domain-aware Tokenizer word segmentation process example

Vectorization

After being processed by a domain aware tokenizer, the research data has been segmented into fine-grained semantic units such as tags, attributes, function names, symbols, and n-grams. However, these tokens are still essentially discrete symbols and cannot be directly used as inputs for lightweight neural networks. They need to be vectorized and mapped to the numerical space so that the neural network model can learn.

The goal of vectorization is to fully preserve the syntactic structure and confusion patterns in XSS attack payloads, enabling the model to recognize attack samples with different manifestations but semantically equivalent (Mahdavifar et al., 2023), while transforming high-dimensional and sparse symbol representations into low dimensional and dense feature vectors, thus meeting the efficient input requirements of lightweight neural networks (LXDNN).

This study designs a TF-IDF (Term Frequency Across Document Frequency) feature representation scheme that combines word level and character level features, targeting the characteristics of XSS attack payloads with fine symbol level features and key semantic level structures. Truncated SVD (Latent Semantic Analysis, LSA) is also introduced to reduce the dimensionality of high-dimensional sparse matrices.

Let the original training dataset be:

$$\mathcal{D} = \{d_1, \dots, d_N\}$$

Each text request d_i has been normalized in Section 4.3.4. Based on this, the domain-aware tokenizer $\varphi(\cdot)$ is used to map each request d_i into a set of fine-grained tokens, which are then converted into vector-label pairs:

$$\mathrm{D} = \{(x_i, \mathrm{y}_i)\ , x_i \in \mathbb{R}^k, y_i \in \{0,1\}, 1 \le \mathrm{i} \le \mathrm{N}\}$$

Where, x_i represents the low-dimensional dense feature vector obtained by combining word-level and character-level TF-IDF representation with Truncated SVD dimensionality reduction. The dimension k is fixed at 200. $\mathrm{y}_i = 1$ indicates the attack class (XSS), and $\mathrm{y}_i = 0$ indicates the normal class.

The vectorization of the training set D is completed in the following three steps:

Step 1: Constructing TF-IDF

Let $f_{\mathrm{t},d}$ be the raw count of token t in document d, document frequency $df_t = \#\{d \in D: f_{\mathrm{t},d} > 0\}$, and use smoothed IDF to get $idf(t) = \log\frac{N+1}{df_t+1} + 1$.

To balance stability and performance, sublinear TF (sublinear_tf) is used. The formula is as follows:

$$tf_{t,d} = \{^{1+\log(f_{t,d}),f_{t,d}>0}_{0,\qquad\quad f_{t,d}=0}$$

For each document d and token t, we calculate $tfidf_{t,d} = tf_{t,d} \cdot\ idf(t)$, and we can get the sparse matrices $X^{\omega} \in \mathbb{R}^{N\times V_{\omega}}$ and $X^{c} \in \mathbb{R}^{N\times V_c}$, where X^{ω} is the token-level term-document matrix and X^{c} is the char-level term-document matrix.

Step 2: Concatenate features

To ensure consistent vector length and stabilize the distance metric, perform L2 normalization on each row:

$$\widetilde{x_d} = \frac{\mathrm{x}_d}{||\mathrm{x}_d||_2}$$

Concatenate the token-level and char-level sparse representations horizontally:

$$X = [X^{\omega}|X^{c}]\ \in \mathbb{R}^{N\times V_{tot}}$$

Step 3: Truncated Singular Value Decomposition (Truncated SVD)

At this stage, the high-dimensional sparse matrix representation of the tokenized dataset, denoted as X, is subjected to Truncated Singular Value Decomposition (Truncated SVD). This dimensionality reduction technique factorizes X into a lower-rank approximation, thereby projecting the original sparse features into a compact latent semantic space while preserving the most informative components. By retaining only the top-k singular values and their corresponding singular vectors, Truncated SVD effectively eliminates noise and redundant correlations, reduces computational overhead, and enhances the efficiency of subsequent model training.

$$X \approx U_k \Sigma_k {V_k}^{\mathsf{T}}$$

Where $U_k \in \mathbb{R}^{N\times k}$、$\Sigma_k \in \mathbb{R}^{k\times k}$、$V_k \in \mathbb{R}^{V_{tot}\times k}$ 、$k \ll V_{tot}$.

The low-dimensional dense representation of a document is:

$$X_k = U_k \Sigma_k \in \mathbb{R}^{N\times k}$$

Equivalently, since $VX_k = U_k\Sigma_k$, for any document sparse representation xxx (row vector), its k-dimensional representation can be calculated as:

$$z = x\mathrm{V}_k \in \mathbb{R}^k$$

Therefore, for the test sample X_{test} it is only necessary to multiply its input vector by the projection matrix V_k obtained in the training phase to achieve a low dimensional dense

representation consistent with the training data. Through this mapping method, this study effectively completes the conversion from domain aware word segmentation results to numerical feature vectors.

Through the collaborative design of layered decoding and domain aware modeling, the method in this paper realizes a complete link from the restoration of confused samples to the expression of semantic features, providing an effective feature basis for cross site scripting attack detection.

Experiment and Result Analysis

Experimental Design and Evaluation Indicators

In order to verify the effectiveness of the proposed layered triggered data preprocessing mechanism and domain aware feature modeling strategy in XSS attack detection, baseline comparison and ablation experiments were conducted from three aspects: model performance, semantic restoration ability and generalization effect.

The experiment was completed on the Ubuntu 20.04 LTS operating system. The CPU is Intel Xeon Gold 6248, the memory is 64GB DDR4, and the storage is 1TB SSD; The development environment is Python 3.9. In order to meet the lightweight target requirements and be consistent with the actual deployment scenario, all experiments do not use GPU acceleration. In order to comprehensively evaluate the performance of the model and ensure the reliability of the results, this paper constructs a dataset for XSS detection tasks. The data set includes attack samples and normal samples. The attack samples mainly come from the open source XSS vulnerability library XSSed and the relevant public projects on GitHub, and the new variant XSS samples that have emerged in recent years have been collected through custom crawlers, ensuring the diversity and timeliness of attack samples. Based on the list of Umbrella top-level domain names ranked top globally by Alexa, random walk algorithm is used to simulate the behavior of real users for crawling to obtain normal URLs and script content close to the actual application scenario. Finally, a data set containing 30000 samples was constructed through uniform random sampling, including 15000 XSS attack samples and 15000 normal samples, which were randomly divided into training sets and test sets according to the ratio of 8:2. All data are cleaned and de duplicated uniformly.

In this study, accuracy, precision, recall, F1 Score and other indicators were used to comprehensively evaluate the detection performance. In order to compare efficiency fairly, all models (including deep learning baseline) are reasoned under the same batch size (batch size=64), and their average throughput (samples/second) is reported as an efficiency indicator.

Baseline Comparison and Ablation Test

In order to comprehensively evaluate the detection performance and module contribution of the method proposed in this paper, three experimental schemes are designed under the same data set and feature configuration:

•Baseline A

Perform basic pre-processing on cleaned samples, decode URL and HTML entities, and then use TF-IDF vectorization and Logistic Regression classification. This scheme represents the traditional lightweight text classification process.

•Baseline B
On the basis of Baseline A, "layered triggered decoding mechanism" is added to realize multi-layer decoding through confusion detection, recursive semantic restoration, pattern compensation and other methods to repair the semantic distortion caused by multiple nesting confusion.
•Proposed
On the basis of Baseline B, we further introduce the "domain aware feature modeling strategy", integrate HTML structural features and JavaScript behavioral features, and build a joint feature representation of cross domain contexts.
The three schemes all use the Logistic Regression classifier and repeat the experiment five times under the same parameters and random seeds to get the average results. Table 1 summarizes the performance and efficiency performance of each scheme.

Table 1
Comparison of performance and efficiency of each method

Scheme	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	throughput (samples/second)
Baseline A	99.22	99.24	98.47	99.21	1250
Baseline B	99.48	99.36	99.14	99.61	667
Proposed	**99.58**	**99.86**	**99.86**	**99.59**	**476**

It can be seen from the results in the table that with the gradual introduction of modules, the detection performance of the model shows a trend of steady improvement. After the introduction of the layered decoding mechanism, the recall rate increased by 0.67 percentage points, indicating that the mechanism can effectively restore the confused malicious samples and reduce missing reports. On this basis, after further introducing domain awareness feature modeling, the accuracy and recall rate reach the highest level of 99.86%, which indicates that the method can more accurately identify malicious patterns while maintaining a high recall rate.

In terms of efficiency, as the complexity of the processing flow increases, the throughput decreases accordingly, which is in line with expectations. However, even the most complex method in this paper, its throughput still reaches 476 samples/second, fully meeting the requirements of high concurrency real-time detection. The comprehensive analysis shows that the strategy of "hierarchical semantic decoding+domain awareness feature fusion" proposed in this paper improves the accuracy and robustness of XSS attack detection at the cost of acceptable computational overhead.

Comparison with Deep Learning Methods
In order to further evaluate the balancing ability of the proposed method between performance and efficiency, this paper compares it with two mainstream deep learning detection models:
• LSTM-ED (Alaoui&Nfaoui, 2023): detection model based on LSTM encoder decoder structure;
• CNN BiLSTM Attn (Li et al., 2022): depth model combining CNN, bidirectional LSTM and attention mechanism.
The comparison results are shown in Table 2. It should be noted that the deep learning method usually requires GPU acceleration to achieve optimal efficiency, but in order to

ensure the fairness of comparison and meet the lightweight deployment scenario, all models in this experiment are run in a pure CPU environment.

Table 2
Performance comparison with deep learning method

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	throughput (samples/second)
LSTM-ED	99.52	99.45	99.38	99.41	115
CNN	99.61	99.88	99.52	99.70	81
Proposed	99.58	99.86	99.86	99.59	476

The experimental results show that the accuracy of the most advanced deep learning model CNN BiLSTM Attn is 99.61%, which has a slight advantage, reflecting the strong feature learning ability of the deep learning model. However, the method proposed in this paper is close to F1 Core, and performs better in recall rate. In terms of reasoning efficiency, the method in this paper also has advantages. Its throughput is 476 samples/s, 4.1 times that of LSTM-ED model, and 5.9 times that of CNN BiLSTM Attn model. This result proves that by improving preprocessing and feature engineering, this paper enables simple linear models to achieve or even surpass part of the performance of complex deep learning models, and at the same time, it improves the computational efficiency by an order of magnitude, providing a better solution for resource constrained practical application scenarios.

Result Analysis and Discussion

The experimental results show that the layered decoding mechanism proposed in this paper plays a key role in improving the recall rate of the model. This mechanism can effectively identify and restore the attack payload that has been confused by multiple codes, restore its original semantic features, and make the original hidden malicious code pattern fully present, thus reducing the missing reports in the detection process.

In feature modeling, the introduction of domain awareness method effectively optimizes the classification boundary. By deeply integrating HTML structural characteristics with JavaScript behavioral characteristics, the model can more accurately capture the essential differences between normal samples and malicious samples in the structure behavior dimension. This improvement enables the model to maintain high detection accuracy while significantly improving the recall rate, and achieves an effective balance between the two types of errors. As a whole, this method achieves a good balance between detection performance and computational efficiency. The potential of the traditional machine learning model has been fully exploited through the refined pretreatment process and targeted feature engineering. This method is close to the complex deep learning model in detection accuracy, while maintaining a significant advantage in reasoning efficiency. This feature makes the scheme particularly suitable for security protection scenarios with strict real-time requirements, such as Web application firewalls, API gateways, etc.

It should be noted that the semantic understanding of script content in the current method is still relatively superficial. Future research directions include: on the premise of maintaining the lightweight characteristics of the system, exploring the introduction of small-scale pre training language model to enhance the ability to understand the deep logic of scripts, so as

to further improve the detection ability of new attacks and advanced threats while maintaining efficient reasoning.

Conclusion

Aiming at the problems of incomplete pre-processing and simplification of feature modeling in XSS attack detection, this paper proposes an improved scheme combining layered triggered decoding and domain aware feature modeling. By designing a detection driven recursive decoding mechanism, the precise semantic restoration of multiple confusion attack payloads is realized; At the same time, a cross domain representation method integrating HTML structural features and JavaScript behavioral features is constructed, which improves the ability of feature discrimination.

Experimental results show that the method in this paper outperforms the traditional pretreatment process in detection performance, especially in the aspect of recall. Compared with the complex deep learning model, this scheme achieves several times of throughput improvement on the premise of maintaining a considerable detection accuracy, showing an excellent balance between detection accuracy and computing efficiency. This provides a practical technical path for security protection in high concurrency scenarios such as Web application firewalls and API gateways.

Based on the current research results, the follow-up work will be carried out from three directions: first, explore the introduction of lightweight pre training model to enhance the depth of understanding of script semantics; The second is to extend this method to other Web attack detection scenarios such as SQL injection; The third is to study privacy computing technologies such as federated learning to ensure model collaborative training and deployment under data security.

References

Alaoui, R. L., & Nfaoui, E. H. (2023). Cross site scripting attack detection approach based on LSTM encoder-decoder and word embeddings. International Journal of Intelligent Systems and Applications in Engineering, 11(2), 277-282.

Alaoui, R. L., & Nfaoui, E. H. (2023). Cross site scripting attack detection approach based on LSTM encoder-decoder and word embeddings. International Journal of Intelligent Systems and Applications in Engineering, 11(2), 277-282.

Alsaffar, M., Aljaloud, S., Mohammed, B. A., Al-Mekhlafi, Z. G., Almurayziq, T. S., Alshammari, G., & Alshammari, A. (2022). Detection of web cross-site scripting (xss) attacks. Electronics, 11(14), 2212.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

Gupta, S., & Gupta, B. B. (2016). XSS-SAFE: a server-side approach to detect and mitigate cross-site scripting (XSS) attacks in JavaScript code. Arabian Journal for Science and Engineering, 41(3), 897-920.

Halko, N., Martinsson, P. G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review, 53(2), 217–288.

Hamzah, K. H., Osman, M. Z., Anthony, T., Ismail, M. A., Abdullah, Z., & Alanda, A. (2024). Comparative analysis of machine learning algorithms for cross-site scripting (XSS) attack detection. JOIV : International Journal on Informatics Visualization, 8(3–2), 1678.

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

Lee, W. M. (2019). Getting started with scikit-learn for machine learning. Python Machine Learning, John Wiley & Sons Inc, 93-117.

Li, Z., Liu, F., Gu, Z., & Liu, Y. (2025). XSS attack detection method based on cnn-bilstm-attention. Applied Sciences, 15(16), 8924.

Mahdavifar, S., & Ghorbani, A. A. (2023). CapsRule: Explainable deep learning for classifying network attacks. IEEE Transactions on Neural Networks and Learning Systems, 35(9), 12434-12448.

Meghdouri, F., Zseby, T., & Iglesias, F. (2018). Analysis of lightweight feature vectors for attack detection in network traffic. Applied Sciences, 8(11), 2196.

OWASP. (2021). OWASP top ten 2021. OWASP Foundation. https://owasp.org/www-project-top-ten/2021

Pazos, J. C., Legare, J. S., Beschastnikh, I., & Aiello, W. (2020). Precise XSS detection and mitigation with Client-side Templates. arXiv e-prints, arXiv-2005.

Pazos, J. C., Legare, J.-S., & Beschastnikh, I. (2021). XSnare: Application-specific client-side cross-site scripting protection. 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 154–165.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

Pradeepa. P. K. (2022). A survey on an investigation of detection & prevention methods for cross-site scripting (XSS) attacks. International Journal of Advanced Research in Science, Communication and Technology, 405–413.

Rathore, S., Sharma, P. K., & Park, J. H. (2017). XSSClassifier: An efficient XSS attack detection approach based on machine learning classifier on snss. Journal of Information Processing Systems.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5), 513–523.

Subba, B., & Gupta, P. (2021). A tfidfvectorizer and singular value decomposition based host intrusion detection system framework for detecting anomalous system processes. Computers & Security, 100, 102084.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., & Attenberg, J. (2009). Feature hashing for large scale multitask learning. Proceedings of the 26th Annual International Conference on Machine Learning, 1113–1120.