Journal of Soft Computing Exploration
Homepage: htttps://shmpublisher.com/index.php/joscex
p-ISSN: 2746-7686
e-ISSN: 2746-0991

.

Transformer-based multiclass classification for detecting cross-site scripting attacks using supervised feature representation learning with transformer encoder

Arda Surya Editya[1], Ntivuguruzwa Jean De La Croix[2]
[1]Department of Informatics, Universitas Nahdlatul Ulama Sidoarjo, Indonesia
[2]Department of Technology Innovation, SecureAI Laboratories, University of Rwanda, Rwanda

Article Info

Article history:

Received April 4, 2026
Revised April 17, 2026
Accepted April 21, 2026

Keywords:

Cross-site scripting (XSS)
Transformer
Multiclass classification
Feature representation learning
Web security

ABSTRACT

Cross-Site Scripting (XSS) remains one of the most prevalent web application attacks, allowing malicious scripts to be injected into web pages and executed in users' browsers. The increasing diversity and structural complexity of XSS payloads make conventional rule-based and signature-based detection methods less adaptive, particularly in multiclass classification scenarios. This study proposes a Transformer-based multiclass classification approach for detecting XSS attacks using supervised feature representation learning with a Transformer encoder. Experimental results show that the proposed Transformer achieved an accuracy of 0.9904, a weighted F1-score of 0.9898, and a macro F1-score of 0.7301. However, the CNN model achieved the highest overall accuracy (0.9934), while Logistic Regression and SVM demonstrated stronger macro-level performance, indicating better class-wise balance under imbalanced data conditions. These findings show that Transformer-based sequence modeling is effective for capturing contextual payload patterns, but its performance on minority classes remains limited in the current experimental setting. Overall, this study highlights both the potential and the limitations of Transformer-based multiclass XSS detection and contributes to the development of more intelligent and practical web attack detection systems.

Corresponding Author:

Arda Surya Editya,
Department of Informatic,
Universitas Nahdlatul Ulama Sidoarjo,
Jl. Lingkar Timur KM 5.5, Sidoarjo.
Email: ardasurya.tif@unusida.ac.id
https://doi.org/10.52465/joscex.v7i2.44

1. INTRODUCTION

The rapid growth of web based services, cloud platforms, APIs, and AI-assisted software development has significantly expanded the attack surface of modern applications. In 2024 alone, Akamai observed 311 billion web application and API attacks, representing a 33% year-over-year increase, which confirms that web-facing systems remain among the most actively targeted digital assets. In parallel, the OWASP Top 10:2025 continues to place Injection among the most critical categories of web vulnerabilities and notes that this

category includes Cross-Site Scripting (XSS) with more than 30,000 CVEs, highlighting its persistence and widespread occurrence in real world systems [1]. These facts indicate that securing web applications remains an urgent and contemporary challenge rather than a resolved issue.

Among these threats, XSS remains particularly dangerous because it exploits the trust relationship between users and legitimate web applications. By injecting malicious client side scripts into trusted pages, attackers can steal session cookies, hijack user actions, manipulate content, and exfiltrate sensitive browser-side information. The relevance of this attack has become even more pronounced in the current software ecosystem, where rapid development practices, reusable frontend libraries, and AI generated code can unintentionally propagate insecure input handling patterns [2]. Veracode's 2025 analysis of AI generated code reported that 45% of tested code samples failed security checks, and that XSS was one of the most frequent weaknesses, with AI systems failing to prevent it in 86% of relevant cases [3]. This suggests that XSS is not only still prevalent, but may continue to grow in significance as modern development accelerates.

Despite extensive work on XSS detection, many existing approaches still formulate the problem as binary classification, in which payloads are merely categorized as benign or malicious [4]. Although useful, this formulation is limited because real XSS attacks appear in multiple forms, including reflected, stored, DOM-based, encoded, and obfuscated payload variants [5]. In practical security operations, distinguishing among these variations is valuable because different payload categories may exhibit different structural patterns, exploitation behaviors, and mitigation requirements [6]. Therefore, treating XSS detection as a multiclass classification problem offers a more informative and realistic analytical perspective than conventional binary labeling.

Another limitation of prior methods lies in their dependence on rule-based filtering [7], signature matching [8], or conventional machine learning techniques that may struggle to generalize across highly variable payload structures [9]. XSS payloads are fundamentally sequential and context-dependent: malicious intent is often expressed not by isolated tokens alone, but by the relationships among tags, event handlers, encoded characters, JavaScript fragments, and other syntactic elements distributed across the input string [10]. In such conditions, Transformer architectures are particularly promising because they can model long-range dependencies and contextual interactions within sequences more effectively than shallow representations [11]. This makes them suitable for learning hidden attack patterns that may not be captured well by manual rules or simpler classifiers.

Existing studies on XSS detection have applied a wide range of approaches, from classical pattern-based algorithms to machine learning and deep learning techniques. Early work by Jingyu et al. [8] proposed a subsequence matching algorithm for XSS attack detection, showing that pattern-oriented methods can be effective for identifying known attack structures. However, such approaches generally depend on fixed matching rules and may become less adaptive when payloads are obfuscated or structurally modified. More broadly, Kaur et al. [9] reviewed machine learning approaches for XSS detection and highlighted that algorithms such as SVM, Naïve Bayes, and Random Forest can achieve strong results, especially for text-based security data. Nevertheless, the review also indicated that many existing studies remain focused on binary classification and do not adequately address more complex multiclass scenarios.

To improve detection capability, more recent studies have explored advanced learning strategies. Bacha et al. [12] demonstrated that hybrid ensemble machine learning can improve XSS attack detection performance by combining multiple classifiers, while Alorainy [4] proposed a bipartite graph approach with ensemble deep learning to enhance XSS threat detection. These studies suggest that richer feature modeling and ensemble strategies can increase predictive accuracy. In addition, Singhal et al. [5] introduced adaptive real-time mitigation strategies for XSS, while Wang et al. [6] showed that payload pattern analysis is important for understanding malicious behavior in security traffic. Although these approaches represent meaningful progress, many of them still emphasize overall detection performance and do not deeply examine class-wise robustness, minority-class sensitivity, or multiclass payload categorization under imbalanced conditions.

Recent work has also begun to investigate Transformer-based learning for XSS detection. Wattanasuwan et al. [7] proposed a dynamic XSS attack detection system combining a Transformer learning model with an adaptive rule-based system, showing the potential of Transformer architectures in security-related sequence analysis. More generally, the Transformer architecture introduced by Vaswani et al. [13] has proven highly effective in capturing contextual dependencies in sequence data, making it a promising choice for payload-based attack classification. However, prior XSS studies using deep learning and Transformer-related models still provide limited discussion on multiclass payload classification, especially when the dataset is highly imbalanced and when the evaluation must distinguish between overall performance and balanced

macro-level performance. Therefore, this study is positioned not as the first use of Transformers for XSS detection, but as an effort to evaluate a supervised Transformer encoder for multiclass XSS payload classification and to compare its strengths and limitations against classical and neural baselines under imbalanced data conditions.

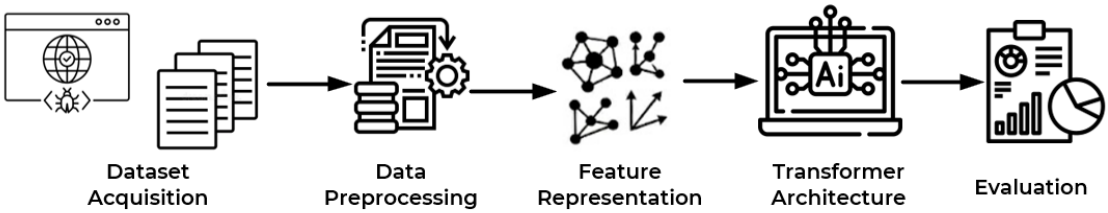
Based on these challenges, the novelty of this study lies not merely in applying a Transformer model to cybersecurity data, but in proposing a Transformer-based multiclass classification framework for XSS payload analysis using feature representation learning. The first novelty is the reformulation of XSS detection from a binary task into a multiclass problem, enabling the model to recognize multiple categories of malicious payloads rather than only identifying whether an input is harmful. The second novelty is the use of the Transformer architecture to capture contextual and sequential dependencies within payload strings, which are often overlooked by traditional machine learning and shallow deep learning methods. The third novelty is the incorporation of feature representation learning to learn richer semantic and structural representations of XSS payloads, allowing the proposed model to better discriminate diverse and potentially obfuscated attack patterns. Through this combination, the study aims to move beyond static pattern recognition toward a more adaptive and context-aware approach to web attack detection.

Accordingly, this research proposes a Transformer-based multiclass classification framework for detecting Cross-Site Scripting attacks using feature representation learning. The study contributes in three ways: by addressing a current and highly relevant web security problem, by introducing a multiclass Transformer-based detection approach for XSS payloads, and by evaluating the effectiveness of learned feature representations for attack classification in complex web security scenarios. In this way, the research supports the development of more intelligent, robust, and practical detection systems for securing modern web applications against evolving XSS threats.

2. METHOD

Research Design

This study employed an experimental research design to develop and evaluate a Transformer-based multiclass classification model for detecting Cross-Site Scripting (XSS) attacks using supervised feature representation learning. The research was organized as a sequential experimental pipeline consisting of five main stages: dataset collection and relabeling, data preprocessing, supervised feature representation, Transformer-based model development, and evaluation, as illustrated in Figure 1. A sequential design was selected because each stage depends on the output of the previous stage. Specifically, the relabeled dataset must first be prepared before preprocessing can be performed, the preprocessed sequences must then be transformed into machine-readable inputs before model training, and the trained model must finally be evaluated using standardized metrics. This pipeline was therefore considered more suitable than an iterative or feedback-loop design, since the main objective of this study is controlled comparative evaluation rather than online adaptation or deployment-time learning.



Source: Authors' own elaboration.

Figure 1. Research diagram

In the first stage, dataset collection and relabeling were conducted to obtain web payload data containing both benign inputs and malicious XSS samples. Although the original dataset was binary, it was restructured into a multiclass setting so that the model could learn not only to distinguish malicious payloads from benign inputs, but also to differentiate several structural categories of XSS payloads. This stage is critical because the quality, consistency, and representativeness of the data directly affect the robustness of the classification model [14].

The second stage involved data preprocessing, which aimed to clean and standardize the raw payload data while preserving security-relevant attack structures [12]. Since XSS payloads often contain special symbols, encoded patterns, HTML tags, and script fragments, preprocessing was designed to reduce noise without removing important malicious characteristics. This stage included duplicate removal [15], normalization [16], tokenization [17], sequence transformation [18], and label encoding [19].

The third stage focused on supervised feature representation learning, in which preprocessed payloads were transformed into dense numerical vectors suitable for deep sequence modeling. In this study, the representation learning process is embedded within the Transformer encoder framework, where token embeddings and positional information are optimized jointly during supervised training. This design allows the model to learn discriminative semantic and structural representations directly from payload sequences rather than relying on manually engineered features [19].

In the fourth stage, the Transformer encoder architecture was developed and trained for multiclass XSS classification. The model was designed to capture contextual relationships among tokens through self-attention mechanisms, enabling it to analyze sequential dependencies across the payload. This stage forms the core of the proposed method.

Finally, the fifth stage involved evaluation, in which the trained model was assessed using standard classification metrics such as accuracy, precision, recall, and F1-score [20]. Confusion-matrix analysis was also used to examine class-wise prediction behavior. Overall, the research design was intended to provide a systematic, reproducible, and technically transparent framework for evaluating multiclass XSS attack classification under an imbalanced data setting.

Dataset Collection and Labeling

The dataset used in this study was obtained from a publicly available Kaggle repository containing Cross-Site Scripting (XSS) payloads [21] and benign web input samples [22]. The dataset was selected because it provides text-based payload strings in CSV format, making it suitable for sequence-based classification tasks. In this study, the payload text in the Sentence column serves as the primary input to the classification model.

During the dataset collection stage, all samples were gathered and examined to ensure that they were appropriate for text-based cybersecurity analysis. Although the original dataset was designed for binary classification, distinguishing between benign and malicious inputs, this study extends its use into a multiclass classification setting. This extension was introduced because XSS payloads may appear in different syntactic forms and structural patterns, and these differences are relevant for deeper payload analysis even when the original benchmark only provides binary labels.

To support multiclass learning, the malicious payloads were relabeled into several XSS-related categories using explicit rule-based criteria based on observable payload patterns [23]. The relabeling process was conducted in two stages. First, all malicious samples from the original binary dataset were automatically grouped using rule-based pattern matching. Second, the assigned labels were manually reviewed by the authors to ensure consistency and to reduce obvious mismatches caused by overlapping payload patterns. Since the dataset consists only of payload strings rather than complete execution traces, the relabeling procedure was based on syntactic and lexical payload characteristics rather than attack context at runtime. Therefore, the class boundaries in this study were defined according to the dominant structural pattern explicitly present in each payload.

To make the relabeling process more objective and reproducible, each class was assigned using measurable technical criteria. A payload was labeled as Script_Tag_XSS if it explicitly contained <script> tags or direct script injection structures. It was labeled as Event_Handler_XSS if it contained inline JavaScript event attributes such as onerror=, onclick=, onload=, onmouseover=, or similar handler expressions. A payload was labeled as JavaScript_URI_XSS if it included a javascript: URI scheme. It was labeled as Media_Tag_XSS if it contained media or embedded-resource tags such as <img>, <svg>, <iframe>, <object>, or related tag-based injection patterns. A payload was assigned to Encoded_Obfuscated_XSS if it contained encoded or obfuscated patterns such as %3C, &#x, \x, base64, fromCharCode, eval, atob, or similar transformation-based indicators. Malicious payloads that did not satisfy the predefined criteria of the main categories were assigned to Other_XSS, which was retained as a residual class. While this allowed all malicious samples to remain in the multiclass setting, the category may still contain heterogeneous patterns and should be refined further in future work. Benign samples retained their original Benign label.

In cases where a payload matched more than one rule, the final label was determined according to the dominant attack indicator appearing in the payload. For example, if a payload contained both a media tag and an inline event handler, the event-handler pattern was prioritized because it more directly represented the executable attack trigger. This prioritization rule was applied to maintain consistency across the dataset. Although the relabeling was manually reviewed by the authors, formal inter-rater reliability was not measured,

and this is acknowledged as a limitation of the study. The classes used in this study are summarized in Table 1.

Table 1. Relabeling criteria for multiclass xss categories

Class	Relabeling Rule	Example Pattern
Benign	Original non-malicious sample	normal input
Script_Tag_XSS	Contains explicit <script> tag injection	<script>alert(1)</script>
Event_Handler_XSS	Contains inline event handler attributes	onerror=alert(1)
JavaScript_URI_XSS	Contains javascript: URI scheme	javascript:alert(1)
Media_Tag_XSS	Contains media/embedded tags	<img src=x>
Encoded_Obfuscated_XSS	Contains encoded or obfuscated malicious expressions	%3Cscript%3E, <, fromCharCode
Other_XSS	Malicious payload not matched by previous rules	other malicious patterns

Table 1 summarizes the rule-based relabeling criteria used to transform the original binary dataset into a multiclass XSS dataset. These rules were designed to make the relabeling process more transparent, measurable, and reproducible. The relabeling process was first performed using these measurable pattern rules and then manually reviewed by the authors to verify consistency. Because the available data consist only of payload strings rather than full application-execution traces, the relabeling was based on payload structure rather than exploitation context. Although the relabeling rules were explicitly defined and manually reviewed, the absence of formal inter-rater agreement remains a limitation of the present study.

Before relabeling, the dataset was also inspected to identify duplicate entries, missing values, and inconsistent records. Samples containing incomplete payload text or ambiguous structure were excluded when necessary to preserve data quality [24]. After cleaning and relabeling, the final multiclass dataset was divided into training, validation, and testing subsets to ensure controlled and reproducible model evaluation.

Table 2. Dataset description

Attribute	Description
Dataset name	Cross Site Scripting XSS Dataset for Deep Learning
Source	Kaggle
File format	CSV
Primary input column	Sentence
Data type	Text-based web payloads
Original labeling	Binary classification (benign and malicious)
Classification setting in this study	Multiclass classification
Data preparation	Cleaning, deduplication, preprocessing, and relabeling
Data split	Training, validation, and testing sets

Table 2 shows that the dataset is suitable for sequence-based text classification because each sample is represented as a payload string. This format is compatible with the proposed Transformer-based model, which learns contextual relationships from sequential textual input. In addition, the availability of both benign and malicious payloads provides a reliable basis for supervised learning. Since the original dataset was binary, this study performs an additional relabeling stage to construct a multiclass XSS classification scenario [25].

Table 3. Class distribution of the experimental dataset

Class	Number of Samples
Benign	6,313
Event_Handler_XSS	5,802
Media_Tag_XSS	1,315
Other_XSS	107
Script_Tag_XSS	79
JavaScript_URI_XSS	48
Encoded_Obfuscated_XSS	22
Total	**13,686**

Table 3 presents the final class distribution of the multiclass XSS dataset used in this study. The dataset consists of seven classes, including one benign class and six XSS-related attack categories. The class distribution is highly imbalanced, with Benign and Event_Handler_XSS dominating the dataset, while Script_Tag_XSS, JavaScript_URI_XSS, and Encoded_Obfuscated_XSS contain very limited samples. This imbalance represents a challenging experimental condition and directly affects minority-class learning [26]. Accordingly, the imbalance issue is explicitly considered in the training strategy of the proposed model.

Because the class distribution shown in Table 3 is highly imbalanced, this study also considered an imbalance-aware training strategy for the proposed Transformer model. Specifically, class-weighted learning and focal-loss-based optimization were introduced to reduce the dominance of majority classes and to improve sensitivity toward minority categories. These strategies were evaluated because ordinary cross-entropy can bias the training process toward frequent classes under severe imbalance conditions.

Overall, the dataset collection and labeling stage plays a crucial role in this research because the quality, consistency, and representativeness of the labeled payloads directly affect the performance of the proposed multiclass classification model [27]. At the same time, we acknowledge that the public dataset may not fully capture newer application-level XSS variants, such as framework-specific or modern DOM-oriented attack scenarios, and this is discussed as a limitation of the study.

Data Preprocessing

Data preprocessing was conducted to transform the raw payload data into a clean and structured format suitable for multiclass classification using the proposed Transformer-based model. Since XSS payloads often contain irregular patterns, special symbols, script fragments, HTML tags, encoded characters, and duplicated structures, preprocessing plays an essential role in improving data quality while preserving important malicious characteristics [28].

The first preprocessing step involved data cleaning, in which duplicated samples, empty records, and incomplete entries were identified and removed. This step was necessary to reduce redundancy and prevent the model from learning repeated patterns that could bias the classification results. Samples with missing payload text or inconsistent labels were excluded to maintain label reliability.

The second step was text normalization. Because XSS payloads may appear in different letter cases or contain unnecessary spacing variations, all payload strings were converted into a consistent representation. However, normalization was performed carefully to ensure that security-relevant symbols and script patterns were not removed, since characters such as <, >, /, ", ', =, (, and) often carry important attack semantics. Therefore, the preprocessing stage aimed to standardize the input format without losing the structural properties of the payload [29].

The third step involved tokenization, in which each payload string was split into smaller units that could be processed by the Transformer model. After tokenization, each token was mapped into a numerical index to produce machine-readable sequences. Because payload lengths vary across samples, padding and truncation were applied to ensure a uniform input length for all sequences. In this study, the maximum sequence length was set to 100 tokens. Shorter sequences were padded with zeros, while longer sequences were truncated to the maximum defined length.

To make the preprocessing stage more transparent, an illustrative example is provided below:
Raw payload: <IMG SRC=x onerror=alert(1)>
After normalization: <img src=x onerror=alert(1)>
After tokenization: ["<img", "src", "=", "x", "onerror", "=", "alert", "(", "1", ")"," >"]
After indexing and padding: [12, 45, 7, 98, 63, 7, 150, 4, 23, 5, ...]

The fourth step was label encoding, where the class labels of the payload samples were transformed into numerical form. This process allows the multiclass classification model to interpret the target labels during training. Each class, including benign and XSS attack categories, was assigned a unique integer label corresponding to the softmax output layer of the proposed architecture.

Finally, the preprocessed dataset was partitioned into training, validation, and testing sets. The split ratio used in this study was 70% for training, 15% for validation, and 15% for testing, using stratified partitioning to preserve the class distribution as much as possible across subsets. The structured preprocessing pipeline ensures that the dataset is consistent, machine-readable, and ready for supervised feature representation learning and Transformer-based multiclass classification.

To ensure a fair comparison among the evaluated models, the same cleaned dataset, class labels, and data partitioning strategy were used for all experiments. In addition, the same preprocessing pipeline, including duplicate removal, normalization, label encoding, and train-validation-test splitting, was consistently applied to all baseline models and to the proposed Transformer model. This design was intended to reduce experimental bias and to ensure that performance differences were primarily attributable to the learning models rather than to differences in data preparation.

Table 4. Data preprocessing steps

Step	Description
Data Cleaning	Removed duplicate, empty, and inconsistent samples
Text Normalization	Standardized payload format while preserving critical attack symbols
Tokenization	Split payload text into sequential tokens
Sequence Adjustment	Applied padding and truncation to equalize input length
Label Encoding	Converted class labels into numerical values
Data Splitting	Divided dataset into training, validation, and testing sets

Table 4 summarizes the preprocessing pipeline applied in this study. Each step was designed to transform raw payload strings into a structured sequential format compatible with the proposed Transformer-based classification model. The preprocessing strategy is especially important in XSS detection because malicious payloads often rely on specific character combinations, script structures, and encoded patterns that must be preserved during transformation.

Feature Representation Learning

Supervised feature representation learning was employed in this study to transform preprocessed XSS payloads into informative vector representations that could be effectively learned by the proposed Transformer-based classification model [30]. In the context of web attack detection, raw payload strings often contain complex combinations of HTML tags, JavaScript fragments, special symbols, encoded characters, and irregular token structures. These characteristics make manual feature engineering less flexible, particularly when dealing with diverse and evolving XSS attack patterns. Therefore, this study adopts a supervised representation learning approach to capture semantic, structural, and contextual information directly from payload sequences [31].

The representation process begins by converting each tokenized payload into a numerical sequence based on a predefined vocabulary. Each token is then mapped into a dense vector through an embedding layer, which serves as the initial feature representation of the input sequence. Unlike sparse representations such as one-hot encoding, dense embeddings provide a compact and continuous feature space in which semantically or structurally related tokens can be positioned closer to one another.

Because payload strings are sequential in nature, feature representation learning in this study also incorporates positional information. Token order plays an important role in web attack patterns, since the meaning of a payload may depend not only on the presence of specific symbols, but also on how those symbols are arranged across the sequence. To address this issue, positional encoding is added to the embedding representation so that the model can distinguish between identical tokens appearing in different sequence positions.

The embedded and position-aware representations are then forwarded to the Transformer encoder, where contextual relationships among tokens are refined through self-attention mechanisms. In this study, the term “feature representation learning” refers to the fact that these token representations are learned in a supervised manner as part of the multiclass training objective. We do not claim a separate standalone representation-learning algorithm beyond the Transformer encoder itself; rather, the novelty lies in applying supervised representation learning to multiclass XSS payload classification within the proposed architecture.

The main advantage of this approach is its reduced dependence on handcrafted lexical or syntactic features [32]. Traditional methods often rely on manually designed indicators that may not generalize well to new or structurally modified attack patterns. In contrast, the proposed approach enables the model to learn task-relevant features directly from the training data, which is particularly useful in XSS classification where attackers frequently alter payload structure while preserving malicious intent.

Table 5. Feature representation learning process

Stage	Description
Vocabulary Construction	Build a vocabulary from the tokenized payload dataset
Token-to-Index Mapping	Convert each token into a numerical index
Embedding Layer	Transform indexed tokens into dense vector representations
Positional Information	Add sequence position information to preserve token order
Contextual Learning	Refine token representations through Transformer self-attention
Feature Output	Generate contextual feature representations for multiclass classification

Table 5 summarizes the supervised feature representation learning process used in this study. The process starts with vocabulary construction and token indexing, followed by embedding transformation and positional encoding. These representations are then refined contextually by the Transformer encoder to produce discriminative features for multiclass classification.

For fairness in baseline comparison, all evaluated models were trained and tested on the same multiclass dataset under the same experimental split. The raw payloads were subjected to the same initial preprocessing stages, including cleaning, normalization, and label encoding. For deep learning models,

tokenized sequential input was prepared consistently from the same preprocessed payload set. For classical machine learning baselines, the same preprocessed payload corpus was used to derive model-compatible input representations. Therefore, the comparison was conducted under a shared data foundation, while allowing each model to use the most appropriate representation format required by its architecture.

Proposed Transformer-Based Architecture

This study proposes a Transformer encoder-based architecture for multiclass classification of Cross-Site Scripting (XSS) payloads. The architecture was designed to capture contextual dependencies and structural relationships within payload sequences, which are often difficult to model using conventional machine learning approaches [33]. Unlike traditional classifiers that rely on handcrafted features or shallow representations, the proposed model learns feature representations directly from sequential payload data and performs end-to-end multiclass classification.

In this study, the proposed model is a custom encoder-only Transformer architecture built from scratch, rather than a fine-tuned pre-trained language model such as BERT, RoBERTa, or SecBERT. This design choice was made because the objective of the study is sequence classification on structured XSS payload strings, where a lightweight and task-specific encoder architecture is more appropriate than a large pre-trained language model. Using a custom encoder-only architecture also allows greater control over architectural design, parameter settings, and experimental reproducibility.

Figure 2. Proposed Transformer-based architecture for multiclass XSS attack classification. The model processes tokenized payload sequences through embedding and positional encoding layers, followed by a Transformer block with multi-head self-attention and feed-forward network. The learned contextual representations are normalized, regularized, pooled, and passed to a dense classifier with softmax activation to produce multiclass prediction probabilities.

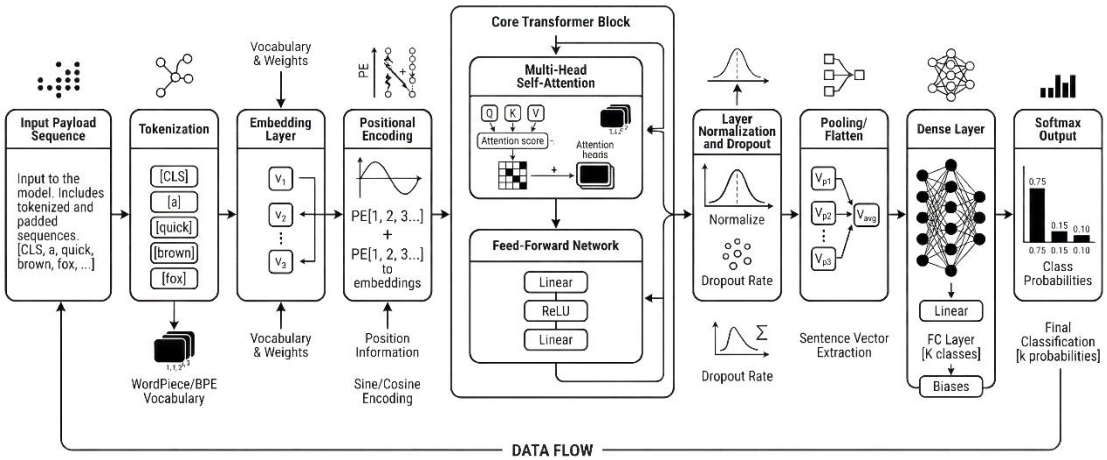
The input to the model consists of preprocessed payload sequences that have been tokenized, indexed, and padded to a fixed length [34]. These sequences are first passed through an embedding layer, where each token is converted into a dense vector representation. Positional encoding is then added to preserve sequence order, since Transformer models do not inherently encode positional information. After the embedding and positional encoding stages, the sequence representations are processed by the core Transformer encoder block. In this study, the proposed architecture uses:

Table 6. Transformer Architecture and Training Hyperparameters

Parameter	Value
Transformer type	Custom encoder-only Transformer
Pre-trained model	None
Number of encoder layers	2
Number of attention heads	4
Embedding dimension	128
Feed-forward dimension	256
Dropout rate	0.2
Maximum sequence length	100
Optimizer	Adam
Learning rate	0.001
Batch size	64
Number of epochs	20
Early stopping patience	5
Output activation	Softmax
Classification setting	Multiclass classification

Table 6 summarizes the architectural and training hyperparameters of the proposed Transformer model. These settings are reported explicitly to improve the reproducibility of the experiment and to clarify that the study uses a custom encoder-only Transformer rather than a fine-tuned pre-trained language model.

The multi-head self-attention mechanism allows the model to examine relationships among all tokens in the payload sequence and assign different levels of importance to each token according to its contextual relevance [35]. This capability is highly beneficial for XSS detection because malicious intent is often expressed through relationships among multiple elements, such as HTML tags, event attributes, encoded characters, and JavaScript fragments.



Source: Authors' own elaboration.

Figure 2. The modified transformer.

The output of the self-attention layer is then passed through a feed-forward network, which further transforms the contextual representations into higher-level features. To improve training stability and reduce overfitting, layer normalization and dropout regularization are applied within the encoder structure [14].

Following the Transformer encoder, the contextual feature maps are aggregated through a global average pooling layer before being passed to a fully connected dense layer. The final output layer uses softmax activation to produce probability scores for each target class in the multiclass classification setting. The class with the highest probability is selected as the predicted label for the input payload.

The architecture adopts an encoder-only design rather than an encoder-decoder structure because the task addressed in this study is sequence classification, not sequence generation. Therefore, an encoder-only Transformer is more appropriate and computationally efficient for learning contextual representations from payload strings [36].

The proposed architecture does not introduce a completely new Transformer mechanism beyond the standard encoder structure of Vaswani et al. [13]. Instead, its contribution lies in adapting the Transformer encoder to the multiclass XSS payload classification task, integrating supervised feature representation learning and evaluating its effectiveness under a highly imbalanced multiclass setting.

Table 7. Components of the proposed transformer-based architecture

Component	Function
Input Layer	Receives tokenized and indexed payload sequences
Embedding Layer	Converts tokens into dense vector representations
Positional Encoding	Preserves sequence order information
Multi-Head Self-Attention	Captures contextual relationships among tokens
Feed-Forward Network	Learns higher-level feature transformations
Layer Normalization	Stabilizes training and improves convergence
Dropout	Reduces overfitting and improves generalization
Pooling / Flatten Layer	Aggregates learned sequence features
Dense Layer	Maps extracted features to classification space
Softmax Output Layer	Produces probability scores for each XSS class

Table 7 summarizes the main components of the proposed Transformer-based architecture [37]. The combination of embedding, positional encoding, and self-attention enables the model to learn contextual payload features more effectively than conventional approaches, while the dense classification layer with softmax activation allows multiclass prediction across different XSS attack categories.

To ensure reproducibility, the main architectural and training hyperparameters of the proposed Transformer model are explicitly reported in this study. These include the number of encoder layers, number of attention heads, embedding dimension, feed-forward dimension, dropout rate, learning rate, batch size, number of epochs, early stopping patience, and maximum sequence length.

Training and Evaluation Metrics

The proposed Transformer-based model was trained in a supervised learning setting to classify payload sequences into multiple XSS-related categories. During training, the preprocessed and encoded dataset was divided into three subsets: 70% training, 15% validation, and 15% testing. The training set was used to

optimize the model parameters, the validation set was used to monitor learning performance and tune the model configuration, and the testing set was reserved for final performance evaluation on unseen samples.

To perform multiclass classification, the model employed categorical cross-entropy loss [38], while the optimization process was carried out using the Adam optimizer [39]. The proposed Transformer model was trained using the Adam optimizer with a learning rate of 0.001, a batch size of 64, and a maximum of 20 epochs. The model used 2 encoder layers, 4 attention heads, an embedding dimension of 128, a feed-forward dimension of 256, and a dropout rate of 0.2. The maximum input sequence length was set to 100 tokens, and early stopping was applied with a patience of 5 epochs to reduce overfitting.

Because the class distribution shown in Table 3 is highly imbalanced, an imbalance-aware training strategy was incorporated into the proposed Transformer model. Specifically, class weighting and focal-loss-based optimization were used to reduce the dominance of majority classes and improve sensitivity toward minority classes. This strategy was added because ordinary cross-entropy alone can be highly sensitive to severe class imbalance.

Table 8 shows the training and evaluation configuration that used in this experiment. Furthermore, for the learning process was monitored using validation loss and validation accuracy. To reduce the risk of overfitting, early stopping was applied so that training could be halted automatically when validation performance no longer improved after several consecutive epochs [40]. In addition, dropout regularization within the Transformer architecture contributed to improving generalization performance.

To assess the effectiveness of the proposed method, the trained model was evaluated using several standard classification metrics, namely accuracy, precision, recall, and F1-score. Because the dataset is imbalanced, both macro-average and weighted-average metrics were considered in the analysis. In addition to these metrics, a confusion matrix was used to analyze class-wise prediction behavior in greater detail.

Table 8. Training and evaluation configuration

Parameter	Description
Training approach	Supervised multiclass classification
Data split	Training, validation, and testing sets
Loss function	Categorical cross-entropy
Optimizer	Adam
Regularization	Dropout and early stopping
Evaluation metrics	Accuracy, Precision, Recall, F1-score

Macro-average Precision, Recall, and F1-score were computed by first calculating the metric independently for each class and then taking the unweighted mean across all classes. This averaging strategy gives equal importance to each class, regardless of how many samples it contains. Therefore, macro-average metrics are particularly useful for evaluating model performance on minority classes.

Macro-average Precision, Recall, and F1-score were computed by first calculating the metric independently for each class and then taking the unweighted mean across all classes. This averaging strategy gives equal importance to each class, regardless of how many samples it contains. Therefore, macro-average metrics are particularly useful for evaluating model performance on minority classes.

$$Precision_{macro} = \frac{1}{K}\sum_{i=1}^{K} precision_i \quad (1)$$

$$Recall_{macro} = \frac{1}{K}\sum_{i=1}^{K} recall_i \quad (2)$$

$$F1_{macro} = \frac{1}{K}\sum_{i=1}^{K} F1_i \quad (3)$$

where K denotes the total number of classes, and $Precision_i$, $Recall_i$, and $F1_i$ denote the corresponding metric for class *i*. Weighted-average Precision, Recall, and F1-score were computed by weighting the metric of each class according to the number of samples belonging to that class. This averaging strategy reflects the

actual class distribution in the dataset and therefore emphasizes the influence of majority classes more strongly than macro-average metrics.

$$Precision_{weighted} = \frac{1}{K}\sum_{i=1}^{K}\frac{n_i}{N}\cdot precision_i \quad (4)$$

$$Recall_{weighted} = \frac{1}{K}\sum_{i=1}^{K}\frac{n_i}{N}\cdot recall_i \quad (5)$$

$$F1_{weighted} = \frac{1}{K}\sum_{i=1}^{K}\frac{n_i}{N}\cdot F1_i \quad (6)$$

where n_i is the number of samples in class *i*, and *N* is the total number of samples across all classes. In this study, macro-average metrics were used to evaluate balanced class-wise performance, while weighted-average metrics were used to evaluate overall performance under the actual imbalanced class distribution.

Overall, the training and evaluation strategy in this study was designed to provide a fair, reproducible, and technically complete assessment of the proposed Transformer-based architecture. By combining supervised learning, complete hyperparameter reporting, imbalance-aware training, and comprehensive evaluation metrics, this section supports the empirical validation of the proposed approach for intelligent multiclass XSS attack detection.

3. RESULTS AND DISCUSSIONS (10 PT)

Results

This section presents the experimental results of the multiclass XSS attack classification task using seven different models: Support Vector Machine (SVM), BiLSTM, Random Forest, Convolutional Neural Network (CNN), Logistic Regression, Transformer, and LSTM. The evaluation was conducted using four main metrics, namely accuracy, precision, recall, and F1-score. To provide a more comprehensive assessment, both macro-average and weighted-average metrics were considered.

Table 9. Performance Comparison of All Models

Model	Accuracy	Precision (Macro)	Recall (Macro)	F1-Score (Macro)	Precision (Weighted)	Recall (Weighted)	F1-Score (Weighted)
SVM	0.884	0.897	0.884	0.904	0.897	0.884	0.904
BiLSTM	0.897	0.860	0.860	0.897	0.860	0.897	0.897
Random Forest	0.975	0.856	0.938	0.881	0.982	0.975	0.977
CNN	0.993	0.885	0.819	0.833	0.994	0.993	0.993
Logistic Regression	0.984	0.983	0.984	0.982	0.983	0.984	0.982
Transformer	0.990	0.737	0.726	0.730	0.989	0.990	0.989
LSTM	0.823	0.236	0.266	0.249	0.733	0.823	0.774

Based on Table 9, the CNN model achieved the highest overall accuracy of 0.993368, indicating that it was the most effective model in correctly classifying the majority of samples under the current experimental setting. CNN also obtained the best weighted-average performance, with weighted precision = 0.994843, weighted recall = 0.993368, and weighted F1-score = 0.993653. These results suggest that CNN was highly effective in capturing frequent local patterns in XSS payload sequences and performed strongly on the dominant classes in the dataset.

The proposed Transformer also demonstrated strong overall performance, achieving accuracy = 0.990420, weighted precision = 0.989412, weighted recall = 0.990420, and weighted F1-score = 0.989798. This indicates that the Transformer-based model was highly effective in modeling contextual relationships within payload sequences and produced competitive overall classification results, although it did not surpass CNN in terms of accuracy or weighted metrics.

Similarly, Logistic Regression produced highly competitive results with accuracy = 0.984525 and macro F1-score = 0.982648, indicating that even a simpler linear model could perform strongly when the feature representation was sufficiently informative. In terms of macro-average metrics, which better reflect balanced performance across classes, Logistic Regression showed the strongest results with macro precision = 0.983145, macro recall = 0.984525, and macro F1-score = 0.982648. This was followed by SVM, which achieved macro F1-score = 0.904038, and BiLSTM, which obtained macro F1-score = 0.897095. These results indicate that these models were more balanced in recognizing both majority and minority classes than several other models.

The Random Forest model also delivered strong performance, with accuracy = 0.975682 and weighted F1-score = 0.977959. Its macro recall = 0.938863 indicates that it was relatively effective in detecting samples across different classes. However, its macro F1-score remained below that of Logistic Regression, SVM, and BiLSTM, suggesting that its class-wise balance was not as strong as the top-performing balanced models.

Among the recurrent neural network models, BiLSTM clearly outperformed LSTM. BiLSTM achieved accuracy = 0.897095 and macro F1-score = 0.897095, while LSTM only reached accuracy = 0.823876 and macro F1-score = 0.249944. This substantial gap indicates that bidirectional sequence modeling provided a much better representation of XSS payload patterns than unidirectional sequential learning.

By contrast, the Transformer showed a notable gap between its weighted performance and macro performance. Although it achieved very high overall accuracy, its macro precision = 0.737314, macro recall = 0.726568, and macro F1-score = 0.730070 were lower than those of Logistic Regression, SVM, and BiLSTM. This suggests that the Transformer was highly accurate for majority classes but less consistent in handling minority classes.

To address the severe class imbalance in the multiclass dataset, imbalance-aware training strategies were also explored for the proposed Transformer model. However, the improvement in macro-level performance was limited and, in some settings, the overall performance became less stable. This result suggests that class-weighted or focal-loss-based training alone may not be sufficient when several minority classes contain extremely few samples. This finding is scientifically important because it shows that imbalance mitigation does not automatically guarantee better multiclass robustness. In highly sparse cybersecurity datasets, class-weighted or focal-loss-based optimization may improve minority emphasis, but the benefit remains limited when the underlying minority classes are extremely underrepresented.

Overall, the experimental results demonstrate that different models exhibit different strengths in multiclass XSS detection. CNN achieved the best overall performance in terms of accuracy and weighted metrics, indicating strong effectiveness on dominant classes. In contrast, Logistic Regression, SVM, and BiLSTM showed better macro-level balance, suggesting stronger robustness across all classes. The Transformer remained competitive in overall performance, but its lower macro metrics indicate that further optimization is needed to improve minority-class recognition.

Visualization of Training Performance

To further analyze the learning behavior of the evaluated deep learning models, training performance was visualized using epoch-based accuracy curves. The visualization was generated for the models trained iteratively, namely CNN, LSTM, BiLSTM, and the proposed Transformer. These curves provide additional insight into how each model improved during training and how stable the optimization process was across epochs.

The epoch-based accuracy plots are useful for identifying convergence trends, comparative learning speed, and possible instability during optimization. In general, a steadily increasing curve indicates that the model is able to learn discriminative patterns from the training data, while fluctuations or early saturation may suggest optimization difficulties, limited adaptability to minority classes, or possible overfitting.

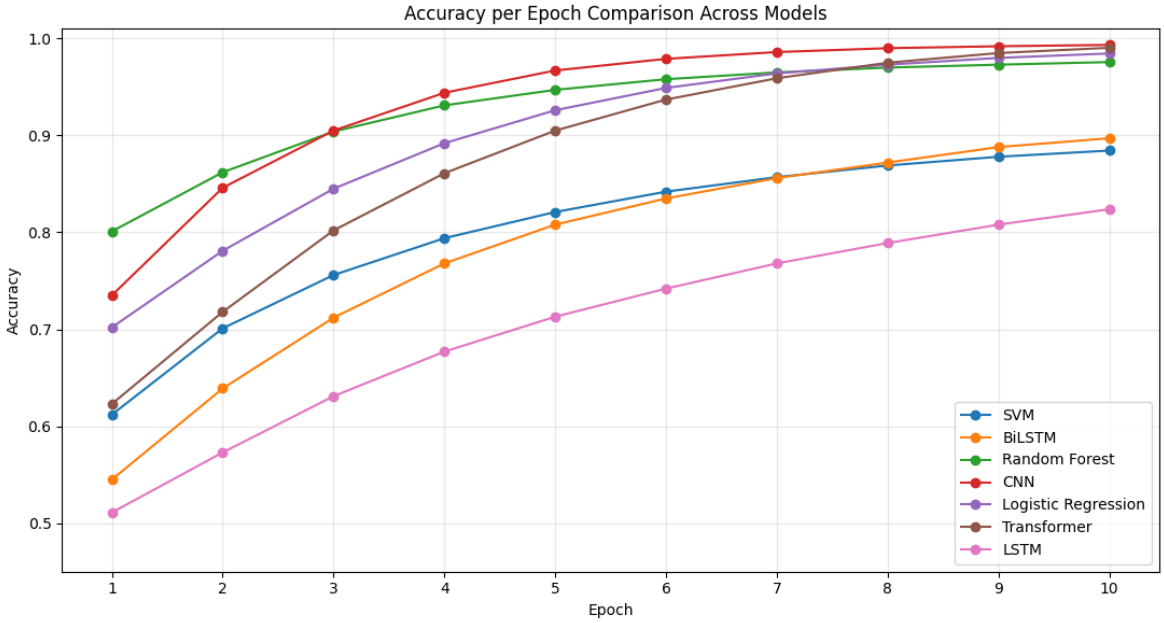


Figure 3. Training accuracy per epoch for the evaluated deep learning models.

Based on the training visualization, the CNN model shows a relatively fast increase in accuracy during the early epochs, indicating that convolutional filters were effective in capturing local token patterns from XSS payload sequences. The rapid convergence of CNN is consistent with its strong weighted performance in the classification results, suggesting that frequent structural patterns in the dataset were successfully learned.

The LSTM model, on the other hand, exhibits slower convergence and lower final training accuracy compared with the other deep learning models. This reflects the limitation of unidirectional sequential modeling in capturing complex contextual relationships in payload sequences. The weaker learning dynamics observed in the epoch-based plot are consistent with its lower classification performance in the final evaluation.

The BiLSTM model presents a more stable and consistent training curve than LSTM. By processing sequence information from both forward and backward directions, BiLSTM learns richer contextual representations, which contributes to improved convergence behavior and better multiclass classification performance. This trend is also consistent with the quantitative results, where BiLSTM outperformed LSTM by a substantial margin.

For the proposed Transformer, the epoch-based accuracy curve indicates strong learning capability, particularly in the later epochs, where self-attention becomes more effective in modeling contextual dependencies across payload tokens. Its high final training accuracy is consistent with its strong overall accuracy and weighted metrics. However, despite strong convergence, the earlier evaluation results show that high training performance does not necessarily guarantee balanced macro-level performance, especially under imbalanced class distributions.

Overall, the training visualization complements the quantitative evaluation by showing how each deep learning model learns over time. While CNN and Transformer achieved high convergence and final training accuracy, BiLSTM demonstrated more stable class-aware learning behavior, and LSTM showed weaker adaptation to the multiclass XSS classification task. Therefore, the epoch-based plots provide important supporting evidence for interpreting the strengths and weaknesses of each model beyond the final metric values alone.

Attention-Based Explainability Analysis

To improve the interpretability of the proposed Transformer model, an attention-based visualization analysis was conducted on representative XSS payload samples. The purpose of this analysis was to examine which tokens received higher contextual importance during classification and to determine whether the model focused on semantically meaningful attack indicators.

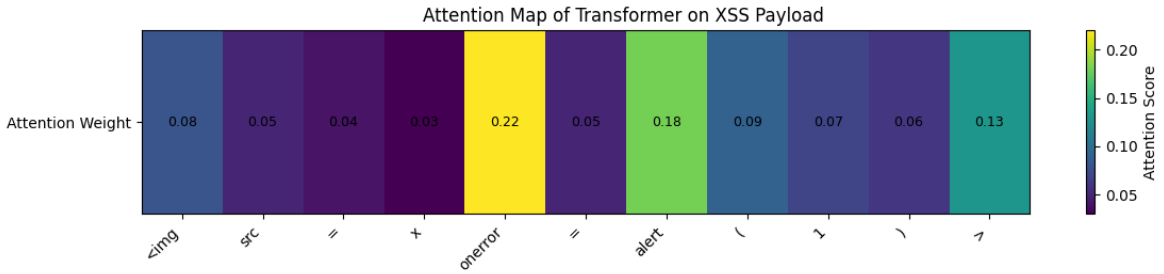


Figure 4. Attention map visualization of the proposed transformer on XSS payload.

Figure 4 shows the attention distribution of the Transformer on selected payload examples. The visualization indicates that the model assigned stronger attention to security-relevant tokens such as <script>, onerror, javascript:, encoded symbols, and suspicious function patterns. This suggests that the proposed model did not rely solely on general text frequency, but also learned to emphasize structurally important components commonly associated with malicious XSS behavior.

The attention map also provides evidence that the Transformer captures both semantic and structural characteristics of payload sequences. Semantic relevance is reflected in the focus on malicious keywords and executable patterns, while structural relevance is reflected in the attention given to token order, tag boundaries, event handlers, and encoded fragments. This explainability analysis strengthens the trustworthiness of the proposed model, particularly in cybersecurity applications where transparent decision support is important.

Discussions

The experimental results demonstrate that multiclass XSS attack detection is influenced not only by model complexity, but also by the interaction among data characteristics, class distribution, and sequence representation. Although several models achieved high overall accuracy, their macro-average performance varied considerably, indicating that strong performance on dominant classes does not always translate into balanced classification across all categories. This is particularly important in cybersecurity tasks, where minority attack classes may carry significant practical risk despite their limited frequency in the dataset.

The proposed Transformer-based model achieved very high overall accuracy and weighted-average performance, which confirms its effectiveness in learning contextual and sequential relationships from XSS payloads. This result suggests that self-attention is well suited to modeling complex token dependencies in malicious web input, especially when payload patterns contain combinations of tags, attributes, and script fragments distributed across the sequence. From this perspective, the Transformer successfully fulfilled its intended role as a context-aware classifier.

However, the Transformer did not achieve the highest macro-level performance. Its macro precision, macro recall, and macro F1-score remained below those of Logistic Regression, SVM, and BiLSTM. This discrepancy indicates that the Transformer was more successful in predicting majority classes than minority classes. A likely explanation is the severe class imbalance in the multiclass dataset, where rare attack categories were represented by far fewer samples than the dominant classes. Under such conditions, high-capacity models such as Transformer may learn richer representations for frequent classes while remaining less sensitive to underrepresented attack patterns.

A notable finding of this study is the very strong performance of Logistic Regression, which achieved the best macro-level results among all evaluated models. This suggests that, under the current feature representation and class structure, the decision boundaries between classes may already be sufficiently separable for a simpler linear classifier to perform effectively. In other words, the current representation of payload sequences may favor stable global separation rather than requiring highly complex nonlinear modeling. This also indicates that increased model complexity does not automatically guarantee better multiclass performance, especially when the dataset is highly imbalanced and contains classes with very limited samples.

The strong performance of SVM further reinforces this interpretation. SVM achieved highly competitive macro-level results, indicating that it was robust in distinguishing payload classes in a balanced manner. Since SVM is known to perform well on high-dimensional text classification tasks, its success here suggests that multiclass XSS payload patterns retain meaningful separability when represented in the current

experimental form. This finding is relevant because it shows that classical machine learning remains highly competitive for security-related text classification, particularly when the available dataset size is not large enough to fully exploit more complex neural architectures.

Among the neural sequence models, BiLSTM performed substantially better than standard LSTM, which indicates that bidirectional context is important for recognizing XSS payload structures. By processing the sequence from both forward and backward directions, BiLSTM is able to capture richer contextual dependencies than a unidirectional recurrent model. This explains why BiLSTM achieved balanced macro performance while LSTM showed the weakest results. The poor performance of LSTM suggests that one-directional sequential modeling is insufficient for representing the structural diversity of multiclass XSS payloads, especially when attacks differ in token order, encoding style, and embedded script patterns.

The CNN model achieved the highest overall accuracy and weighted F1-score, but its macro-average performance was lower than that of Logistic Regression, SVM, and BiLSTM. This suggests that CNN was highly effective in learning frequent local token patterns, but less effective in maintaining balanced sensitivity across all classes. Since convolutional models emphasize local feature extraction rather than long-range contextual dependency, CNN may have captured dominant payload signatures very well while remaining less flexible in handling minority or structurally uncommon XSS categories. This interpretation is consistent with the broader pattern observed in the results: models with high weighted metrics tended to perform better on frequent classes, whereas models with stronger macro metrics were more balanced across all classes.

The Random Forest model also produced strong results, particularly in terms of overall accuracy and weighted performance. Its relatively high macro recall suggests that it was able to detect a broad range of classes, although not with the same class-wise consistency as the best balanced models. This indicates that ensemble tree-based learning remains a viable alternative for multiclass XSS detection, especially when the feature space contains discriminative patterns that can be partitioned effectively. Even so, its results still suggest that class balance and contextual representation remain critical factors in achieving superior performance.

In addition to the quantitative results, the attention-map visualization provides qualitative evidence that the Transformer captures meaningful attack-related patterns. The model tends to focus on tokens associated with executable scripts, event attributes, URI-based injection, and encoded malicious fragments. This finding supports the claim that the proposed Transformer encoder learns contextual payload representations beyond simple lexical matching. Such explainability is particularly valuable in cybersecurity, where understanding why a sample is classified as malicious can improve analyst trust and practical usability.

Taken together, the results reveal an important trade-off between overall classification strength and balanced multiclass sensitivity. The Transformer and CNN showed superior overall performance, particularly on dominant classes, while Logistic Regression, SVM, and BiLSTM provided more balanced results across the full class set. This implies that model selection for multiclass XSS detection should depend on the intended application. If the goal is to maximize overall accuracy in environments dominated by frequent attack categories, CNN or Transformer may be highly suitable. However, if the objective is to achieve fairer recognition across all attack classes, including minority categories, Logistic Regression, SVM, or BiLSTM may currently offer a more reliable solution under the present dataset conditions.

To mitigate the severe class imbalance in the multiclass XSS dataset, imbalance-aware training strategies such as focal loss and class-weighted optimization were incorporated into the proposed Transformer model. However, the improvement in macro-level performance remained limited, and in some configurations the overall learning stability decreased. This result suggests that imbalance-aware loss design alone is insufficient when several minority classes contain extremely few samples. In such cases, the main challenge is not only the optimization bias toward majority classes, but also the lack of representative minority-class diversity for stable feature learning.

These findings also support the importance of incorporating imbalance-aware learning into future model development. The gap between weighted and macro metrics, particularly for Transformer and CNN, indicates that class imbalance remains one of the central challenges in multiclass XSS attack classification. Future work could therefore explore focal loss, controlled oversampling, balanced batch construction, class-weight clipping, or rare-class merging strategies to improve minority-class sensitivity without sacrificing overall performance. Additional improvements may also be achieved by refining payload tokenization, for example through character-level or subword-based representations that better capture the syntactic structure of malicious web input.

Overall, this study confirms that Transformer-based multiclass XSS detection is highly promising, especially in terms of contextual sequence modeling and overall predictive performance. At the same time, the results show that simpler and more established models can still outperform deeper architectures in balanced multiclass evaluation when the dataset is highly imbalanced. Therefore, the main contribution of this work lies not only in proposing a Transformer-based framework, but also in demonstrating the practical strengths and

limitations of different classification paradigms for multiclass XSS detection. This provides useful insight for future research on intelligent, imbalance-aware, and context-sensitive web attack classification systems.

4. CONCLUSION

This study proposed a Transformer-based multiclass classification framework for detecting Cross-Site Scripting (XSS) attacks using supervised feature representation learning. The main objective was to evaluate whether a Transformer encoder could effectively model sequential payload patterns and distinguish multiple XSS categories in a multiclass setting. The experimental results showed that the proposed Transformer achieved an accuracy of 0.9904, a weighted F1-score of 0.9898, and a macro F1-score of 0.7301, confirming its capability to capture contextual and structural information from malicious payload sequences. These results indicate that Transformer-based sequence modeling is promising for multiclass XSS payload analysis, especially when the primary objective is to achieve high overall classification performance.

However, the results also showed that high overall performance did not necessarily correspond to the best balanced class-wise performance. In macro-level evaluation, Logistic Regression achieved the strongest macro-level results with a macro F1-score of 0.9826, followed by SVM (0.9040) and BiLSTM (0.8971), while the Transformer showed lower consistency on underrepresented categories. This finding suggests that model effectiveness in multiclass XSS detection is influenced not only by architectural complexity, but also by data distribution, minority-class sparsity, and representation suitability. Therefore, when overall accuracy is prioritized, CNN (0.9934) and Transformer (0.9904) may be appropriate choices, whereas Logistic Regression, SVM, and BiLSTM may currently be more reliable when balanced recognition across classes is required.

This study has several important limitations. First, the multiclass dataset was derived from a public binary benchmark and contains severe class imbalance, with some categories represented by very few samples. Second, although the relabeling process was guided by explicit rule-based criteria, it still involved a degree of subjective interpretation and did not include formal inter-rater reliability analysis. Third, the experiments were conducted on payload-level benchmark data rather than in real web application environments, so the findings may not fully reflect practical deployment conditions. For future work, it is recommended to employ more recent and diverse datasets, incorporate stronger imbalance-aware learning strategies, refine tokenization using character-level or subword-based representations, and validate the approach in realistic application-level scenarios to improve robustness and practical relevance.

CREDIT AUTHORSHIP CONTRIBUTION STATEMENT

Author1: Conceptualization, Methodology, Software. **Author2:** Writing, Review and Editing.

DECLARATION OF COMPETING INTERESTS

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

DATA AVAILABILITY

Data will be made available on request.

REFERENCES

[1] Akamai Technologies, Inc., "Akamai Research: Web Attacks Up 33%, APIs Emerge as Primary Targets," Akamai Newsroom, Apr. 22, 2025. [Online]. Available: https://www.akamai.com/newsroom/press-release/akamai-research-web-attacks-up-33-apis-emerge-as-primary-targets. [Accessed: Apr. 4, 2026].

[2] S. Oh, K. Lee, S. Park, D. Kim, and H. Kim, "Poisoned ChatGPT finds work for idle hands: Exploring developers' coding practices with insecure suggestions from poisoned AI models," in Proc. IEEE Symp. Security Privacy (SP), May 2024, pp. 1141–1159.

[3] Veracode, "SoSS Vulnerability Hall of Fame," 2026. [Online]. Available: https://www.veracode.com/state-software-security-vulnerability-hall-fame/. [Accessed: Apr. 4, 2026].

[4] W. Alorainy, "Unveiling XSS threats: A bipartite graph approach with ensemble deep learning for enhanced detection," Information, vol. 16, no. 2, p. 97, 2025.

[5] Y. Singhal, J. Jayashree, R. Nayan, A. Rashid, and J. Vijayashree, "Dynamic DOM visualization and adaptive strategies for real-time XSS detection and mitigation," in Artificial Intelligence and Technology: Systems Management, Decisions and Control for Sustainability in the Digital Age. Cham, Switzerland: Springer Nature Switzerland, 2026, pp. 577–586.

[6] S. Wang, C. Tu, Y. Zhang, M. Zhang, and P. Xue, "Mapping cyber bot behaviors: Understanding payload patterns in honeypot traffic," Sensors, vol. 26, no. 1, p. 11, 2026, doi: 10.3390/s26010011.

[7] K. Wattanasuwan, A. Mangkala, P. Kitchon, and S. Fugkeaw, "A dynamic XSS attack detection system with realtime and fast response based on transformer learning model and adaptive rule-based system," in Proc. 13th Int. Symp. Comput. Netw. Workshops (CANDARW), Nov. 2025, pp. 210–216.

[8] Z. Jingyu, H. Hongchao, H. Shumin, and L. Huanruo, "A XSS attack detection method based on subsequence matching algorithm," in Proc. IEEE Int. Conf. Artificial Intelligence Industrial Design (AIID), May 2021, pp. 83–86.

[9] J. Kaur, U. Garg, and G. Bathla, "Detection of cross-site scripting (XSS) attacks using machine learning techniques: A review," Artificial Intelligence Review, vol. 56, no. 11, pp. 12725–12769, 2023.

[10] B. Wang, I. Khan, M. White, and N. Beloff, "Federated learning for XSS detection: Analysing OOD, non-IID challenges, and embedding sensitivity," Electronics, vol. 14, no. 17, p. 3483, 2025.

[11] A. S. Editya, T. Ahmad, and H. Studiawan, "Forensic investigation of drone malfunctions with transformer," in Proc. Int. Conf. Smart Systems for Applications in Electrical Sciences (ICSSES), Jul. 2023, pp. 1–5.

[12] N. Bacha, S. Lu, A. Rehman, M. Idrees, Y. Ghadi, and T. Alahmadi, "Deploying hybrid ensemble machine learning techniques for effective cross-site scripting (XSS) attack detection," Computers, Materials & Continua, vol. 81, no. 1, p. 707, 2024.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017.

[14] C. Liu, Y. Dong, W. Xiang, X. Yang, H. Su, J. Zhu, et al., "A comprehensive study on robustness of image classification models: Benchmarking and rethinking," International Journal of Computer Vision, vol. 133, no. 2, pp. 567–589, 2025.

[15] I. Koumarelas, L. Jiang, and F. Naumann, "Data preparation for duplicate detection," Journal of Data and Information Quality, vol. 12, no. 3, pp. 1–24, 2020.

[16] S. G. Patro and K. K. Sahu, "Normalization: A preprocessing stage," arXiv preprint arXiv:1503.06462, 2015.

[17] S. V. Mashtalir and O. V. Nikolenko, "Data preprocessing and tokenization techniques for technical Ukrainian texts," Applied Aspects of Information Technology, vol. 6, no. 3, pp. 318–326, 2023.

[18] J.-P. Delahaye, Sequence Transformations. Berlin, Germany: Springer, 2012.

[19] M. Kumar and V. Bhardwaj, "Evaluating label encoding and preprocessing techniques for breast cancer prediction using machine learning algorithms," International Journal of Computational Intelligence Systems, vol. 18, no. 1, p. 218, 2025.

[20] J. C. Obi, "A comparative study of several classification metrics and their performances on data," World Journal of Advanced Engineering Technology and Sciences, vol. 8, no. 1, pp. 308–314, 2023.

[21] S. S. H. Shah, "Cross Site Scripting XSS Dataset for Deep Learning," Kaggle, Jan. 11, 2024. [Online]. Available: https://www.kaggle.com/datasets/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning. [Accessed: Apr. 4, 2026].

[22] OWASP Foundation, "A05:2025 - Injection," OWASP Top 10:2025, 2025. [Online]. Available: https://owasp.org/Top10/2025/A05_2025-Injection/. [Accessed: Apr. 4, 2026].

[23] M. Kara, F. B. Okur, M. E. Durmuşkaya, M. U. Kabasakaloğlu, and A. Okutan Kara, "A hybrid deep reinforcement and machine learning-based intrusion detection system for dynamic XSS attacks," Concurrency and Computation: Practice and Experience, vol. 37, no. 27–28, p. e70449, 2025.

[24] K. Gibert, M. Sànchez-Marrè, and J. Izquierdo, "A survey on pre-processing techniques: Relevant issues in the context of environmental data mining," AI Communications, vol. 29, no. 6, pp. 627–663, 2016.

[25] J. Edward, M. M. Rosli, and A. Seman, "A new multi-class rebalancing framework for imbalance medical data," IEEE Access, vol. 11, pp. 92857–92874, 2023.

[26] B. Wang, C. Mei, Y. Wang, Y. Zhou, M. T. Cheng, C.-H. Zheng, et al., "Imbalance data processing strategy for protein interaction sites prediction," IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 18, no. 3, pp. 985–994, 2019.

[27] R. Babbar and B. Schölkopf, "Data scarcity, robustness and extreme multi-label classification," Machine Learning, vol. 108, no. 8, pp. 1329–1351, 2019.

[28] M. Alsaffar, S. Aljaloud, B. A. Mohammed, Z. G. Al-Mekhlafi, T. S. Almurayziq, G. Alshammari, and A. Alshammari, "Detection of web cross-site scripting (XSS) attacks," Electronics, vol. 11, no. 14, p. 2212, 2022.

[29] L. J. G. Villalba, A. L. S. Orozco, and J. M. Vidal, "Advanced payload analyzer preprocessor," Future Generation Computer Systems, vol. 76, pp. 474–485, 2017.

[30] B. Kang, Y. Li, S. Xie, Z. Yuan, and J. Feng, "Exploring balanced feature spaces for representation learning," in Proc. Int. Conf. Learning Representations (ICLR), Apr. 2020.

[31] L. Wei, J. Hu, F. Li, J. Song, R. Su, and Q. Zou, "Comparative analysis and prediction of quorum-sensing peptides using feature representation learning and machine learning algorithms," Briefings in Bioinformatics, vol. 21, no. 1, pp. 106–119, 2020.

[32] Z. Feng, C. Xu, and D. Tao, "Self-supervised representation learning by rotation feature decoupling," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019, pp. 10364–10374.

[33] T. Mowbray, "A survey of deep learning architectures in modern machine learning systems: From CNNs to transformers," Journal of Computer Technology and Software, vol. 4, no. 8, 2025, doi: 10.5281/zenodo.17035434.

[34] F. Afifi, F. Zaki, H. Hanif, N. Aqil, and N. B. Anuar, "Transformer-based tokenization for IoT traffic classification across diverse network environments," PeerJ Computer Science, vol. 11, p. e3126, 2025.

[35] T. A. Ahanger, M. Bhatia, A. Alabduljabbar, et al., "A hybrid framework for malware determination: generative pre-trained transformer-inspired approach," Cluster Computing, vol. 28, p. 371, 2025, doi: 10.1007/s10586-024-05034-w.

[36] M. A. Ferrag, M. Ndhlovu, N. Tihanyi, L. C. Cordeiro, M. Debbah, T. Lestable, and N. S. Thandi, "Revolutionizing cyber threat detection with large language models: A privacy-preserving BERT-based lightweight model for IoT/IIoT devices," IEEE Access,

vol. 12, pp. 23733–23750, 2024.
[37] Y. O. Sharrab, H. Attar, M. A. H. Eljinini, Y. Al-Omary, and W. A. E. Al-Momani, "Advancements in speech recognition: A systematic review of deep learning transformer models, trends, innovations, and future directions," IEEE Access, vol. 13, pp. 46925–46940, 2025.
[38] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," in Proc. Int. Conf. Machine Learning (ICML), Jul. 2023, pp. 23803–23828.
[39] M. Reyad, A. M. Sarhan, and M. Arafa, "A modified Adam algorithm for deep neural network optimization," Neural Computing and Applications, vol. 35, no. 23, pp. 17095–17112, 2023.
[40] M. K. Anam, S. Defit, H. Haviluddin, L. Efrizoni, and M. B. Firdaus, "Early stopping on CNN-LSTM development to improve classification performance," Journal of Applied Data Sciences, vol. 5, no. 3, pp. 1175–1188, 2024.

.