Feature Fusion of Pattern, Statistical, and Contextual Representations using Deep and Machine Learning for XSS Detection

Ali Nafea Yousif [1,2], Ziyad Tariq Mustafa Al-Ta'i[1], Widad Ismail[3] and Hassan Muayad Ibrahim[2,4]
[1]*Department of Computer Science, College of Science, University of Diyala, 32001 Baqubah, Iraq*
[2]*University of Information Technology and Communications, 10001 Baghdad, Iraq*
[3]*Auto-ID Laboratory (AIDL), School of Electrical and Electronic Engineering, Universiti Sains Malaysia, 14300 Nibong Tebal, Malaysia*
[4]*School of Electrical and Electronic Engineering, Engineering Campus, Universiti Sains Malaysia, 14300 Nibong Tebal, Pulau Pinang, Malaysia*
scicomphd232404@uodiyala.edu.iq, ziyad1964tariq@uodiyala.edu.iq, eewidad@usm.my, hassan.m@student.usm.my

Keyword: Cross-Site Scripting (XSS), Feature Fusion, Deep Learning, Machine Learning.

Abstract: Cross-Site Scripting (XSS) is still a crucial threat to web applications, enabling attackers to inject malicious scripts and compromise user security. Conventional detection methods encounter scalability and adaptability issues, while recent machine and deep learning approaches often rely on isolated features or single classifiers. This study proposes two approaches to developing XSS detection models. The first model applies an early fusion approach; it combines features derived from handcrafted patterns, statistical representations built using term frequency–inverse document frequency (TF-IDF), and sequential embeddings produced through long-short-term memory (LSTM) and gated recurrent unit (GRU) networks. The second model, based on a late feature fusion approach, extracts multiple feature types, including statistical representations from TF-IDF, handcrafted pattern-based indicators, and sequential embeddings obtained from LSTM and GRU networks, in the next stage, the concatenated feature vectors are then used to train seven well-known machine learning classifiers, including Random Forest, Support Vector Machine, AdaBoost, Decision Tree, Logistic Regression, Gaussian Naïve Bayes, and Multi-Layer Perceptron. A nearly balanced dataset of 13,686 payloads was used. The results show that early fusion achieved the best performance (LSTM+TFIDF: 99.90% accuracy, F1-score 99.91%), while late fusion (GRU+TFIDF+MLP) achieved 99.85% accuracy and an F1-score of 99.86%. These results demonstrate that early fusion improves detection sensitivity, while late fusion enables interpretability and synthesis, providing practical insights for designing robust XSS detection systems.

1 INTRODUCTION

In the latest decades, there has been quick development of web technologies that turned the internet into an interactive medium from the former static information source and made real-time communication, electronic payment, and customized user experience possible. Websites are nowadays very interconnected in key industries like e-commerce, finance, health care, and social networking and deliver services for millions of activities and tasks to the end users on a day-to-day basis [1], [2]. Even though this growth has offered greater access and capability, it has also created a broader attack surface that attackers are able to exploit [3], [4]. XSS attacks have been one of the most influential and still dominate web security threats [5], by which the attackers can inject and run malicious scripts on trusted domains [6]. XSS attacks have long been among the top web security risks [7]. Attackers place malicious scripts in the system to hijack user sessions, steal login credentials, and even spread malware [8], and cause severe economic losses to major industries like e-commerce and online finance [9]. Based on these stolen cookies, the intruder can easily access other Web users' applications and get valuable information [10]. DOM-based XSS, stored-XSS, and reflected-XSS are the three kinds of XSS attacks that can be identified over the web [11]. DOM-based XSS occurs when malicious code is executed on the client side due to insecure manipulation of the DOM. In this type of attack, the injected JavaScript payload is embedded in a crafted URL or input, and executed directly by

the browser without being sent to the server [12]. On sites like forums and blogs, stored-XSS vulnerabilities are easier to find, where malicious payloads are directly injected into website inputs and stored in the websites' databases, here the attack is triggered when the victim browses these items [13]. Reflected XSS is the most common type in web applications, it occurs when malicious input provided in an HTTP request is immediately reflected back in the HTTP response without proper validation or sanitization, causing the injected script to execute in the client's browser [14]. Thus, the attacker believes that there is a chance to execute the harmful scripts that are running in the web application, and does so on the client-side to retrieve the victim's details [15]. In recent years, artificial intelligence (AI) techniques have attracted considerable attention, particularly machine learning and deep learning techniques. When compared with conventional methods such as input validation, static analysis, and dynamic analysis, modern AI-based techniques tend to resolve many of the long-standing weaknesses of these traditional approaches and generally provide better detection and adaptability [16], [17].

Nevertheless, identifying XSS attacks is still far from trivial. Common issues such as dependence on single-type classifiers, the limited size of available datasets, class imbalance, and the difficulty of explaining model decisions continue to restrict performance [18].

Most previous studies have concentrated on either sequential representations or statistical features alone, which makes it hard for their models to fully capture both the structural layout and contextual intent of attack payloads.

In this work, we explore two complementary directions to overcome these gaps. The first combines pattern-based and statistical features with deep sequential models (LSTM and GRU) so that structural irregularities and contextual links inside payloads can be learned jointly. The second approach employs deep learning models mainly as feature extractors; the obtained representations are then classified using traditional algorithms such as Random Forest, AdaBoost, Decision Tree, Logistic Regression, Gaussian NB, SVM, and MLP.

The following summarizes the primary contributions to the proposed work:

1) The paper introduces two complementary models that combine machine learning and deep learning techniques for detecting XSS attacks. Each model is developed from a distinctive viewpoint, presenting a different architectural perspective, yet validated utilizing the same experimental environment.
2) The two approaches were examined on the identical benchmark dataset in the same setting for experimentation. This enables a fair and reasonable comparison between the two, deciding the method that is more reliable in practical detection conditions.
3) The obtained results demonstrate consistently high detection accuracy and robustness against various types of attack payloads, showing that both approaches can generalize well across the dataset.

The rest of this paper is structured as follow: Section 2 reviews recent research on XSS detection and discusses the remaining gaps and open challenges in the field. Section 3 describes the proposed methodology, detailing the two designed models and their integration process. Section 4 presents the experimental setup and the obtained results. Section 5 provides a discussion of the findings and their implications. Finally, Section 6 concludes the paper and outlines directions for future research.

2 RELATED WORKS

In this section, several previous studies were reviewed, Table 1 presents the results and details, that adopted different methods and methodologies in detecting Cross-Site Scripting attacks were reviewed to support and gather ideas in applying a methodology that provides better future results.

Muralitharan Krishnan et al. (2024) [19] used a Hybrid Stacking Ensemble approach that combined machine learning algorithms (NB, SVM, k-NN) and deep models such as RNN, CNN, and LSTM with Logistic Regression as the final classifier, in addition to a processing layer based on URL Encoding + Dictionary Mapping to enhance detection and defense against XSS attacks. They used four size sets of 3k to 51k samples from GitHub and Kaggle datasets and attained more than 99.5% accuracy for detection and prevention tasks. This indicates the success of heterogeneous model fusion and proposes the potential to investigate more adaptive fusion models or deeper linguistic embeddings to improve contextual coherence across feature sources.

Wang et al. (2024) [20] proposed the DoubleR framework based on bidirectional detection with the incorporation of request and response analysis for discovering what they have identified as the "attack reality" of XSS attacks. Bagging-GBDT was utilized

after feature construction from the requests and from the responses. Experiments were conducted on five web applications with 11 registered vulnerabilities (CVEs), a training dataset containing 5,872 malicious requests and 7,343 benign requests, and approximately 15,000 response samples. The results demonstrated the ability to distinguish between failed attempts and actual attacks achieving 94% Recall and 86% Precision. Despite the novelty of the approach in linking the response to the request, its heavy reliance on manually generated features and traditional algorithms may limit its generalization to new and complex patterns.

Tadhani et al. (2024) [21] proposed a hybrid model combining CNN and LSTM after decoding, unification, and tokenization to process HTTP requests and payloads. The goal is to detect both SQLi and XSS attacks within a complementry approach. They trained the model on three datasets (HTTP CSIC-2010, the payloads set from Kaggle, and a test set created using Burp) and achieved high accuracy of approximately 99.2–99.8% on the learning models used. The methodology relies on word2vec representation and feature extraction via CNN layers, followed by LSTM to capture sequential consequences. However, their research lacks real-time/production testing and does not provide detailed handling of class imbalance or resistance to obfuscation attacks, leaving room for improvement in generalization and reliability.

Yixian Liu et al. (2025) [22] introduced a parallel architecture combining CNN, LSTM, and a multi-attention Transformer with a final AdaBoost classifier based on sequential and local feature extraction from payloads after processing them via Word2Vec and specific filtering and segmentation of HTML and JavaScript elements. The model was trained on Dataset 1 (from OWASP, PortSwigger, and XSS Cheat Sheet) and Dataset 2 (from DMOZ and XSSed), achieving accuracy of approximately 99.7% and 99.5%, respectively. Although the approach is effective in combining syntactic and contextual patterns of attacks across multiple branches, performance remains more dependent on the structure of individual channels than on a unified representation of the attack context.

Zhou et al. (2025) [23] applied the XSShield methodology, which relies on large language models (LLMs such as GPT-4) to understand the semantic context of JavaScript code to detect stored XSS attacks. The model was evaluated on 610 BeEF attack samples (with obfuscation-disguised versions) and 305 benign JS150k samples, achieving 93% accuracy (F1≈0.9266) on GPT-4 with an average processing time of only 0.205 seconds per file. This approach also highlights the advantage of LLM-based semantic interpretation and prospects for its integration with deep learning or combined models for improved identification of complicated scenarios or low-signature attack scenarios.

Li et al. (2025) [24] suggested a hybrid model based on an integration of CNN, BiLSTM, and a multi-headed attention mechanism for XSS attack identification. Script requests were transformed into vectors by a better segmentation algorithm, and information was passed to model layers for patterns and contextual dependency identification. The model, based on two data sets (XSSed-DMOZ and Merwani-XSS), reached high performance, accuracy of 99.38%, and an F1-score of 99.37%. Although the approach is effective in combining local features and long dependencies, the heavy reliance on static word representations makes sensitivity to shifting patterns partially limited, leaving room for expansion towards deeper dynamic processing.

Xu et al. (2025) [25] presented an interactive approach based on reinforcement learning (RL) using the Hierarchical Multi-Objective Reward-Enhanced Dueling Double (DQN) model to automatically detect XSS vulnerabilities in a simulated attack environment. The system relies on dynamically generating base and mutated payloads, semantic similarity analysis to filter out duplicates, and optimizing the learning process via multi-level rewards (extrinsic + intrinsic + N-Step Rewards). The model was tested on 10 real, open-source web applications (e.g., OWASP Benchmark, DVWA, WAVSEP, etc.) and successfully detected 347 vulnerabilities in ~7789 seconds with an F1-score approximately 95.4%. This approach provides support and confirmation of the importance of adaptive learning and its significant role in enabling the integration of deep learning techniques or context-aware features to support responses to atypical attacks.

Abouda (2025) [26] developed a hybrid approach to detecting XSS by combining natural language processing (NLP) and reinforcement learning (RL) techniques. Text inputs were transformed into standard representations using TF-IDF after multiple cleaning steps such as lemmatization and common word removal. The study relied on training a deep neural network (DNN) on public Kaggle datasets containing approximately 1,300 samples, divided between benign and malicious samples. The study summarized promising results on the training and validation sets, although test performance was not explicitly presented. Although good results are

reported in this study, the reliance on only shallow representations (TF-IDF) and dense networks remains a limiting factor in the ability to capture complex or hidden contextual patterns within payloads, leaving room for improving models towards more context-aware representations.

The reviewed papers indicate the usefulness of ML- and DL-based structures for detecting XSS, but the majority have employed one of these techniques in isolation from another, without considering various feature characteristics. This has restricted their ability to fully capture attack behaviors. To address this, the current paper proposes two complementary approaches: early fusion, which combines handcrafted pattern-based statistical features with sequential deep features, and late fusion, in which deep learning merges both feature types prior to classification by ML algorithms. These models are implemented under consistent experimental settings, providing an appropriate basis for comparison.

3 METHODOLOGIES

3.1 Overview of the Proposed Framework

This paper presents a dual-framework strategy for improving the detection of Cross-Site Scripting (XSS) attacks by combining the strengths of deep learning with classical machine learning techniques. Because no single model provides an optimal solution, this technique is built around the fact that deep learning excels at capturing complicated sequential dependencies and semantic patterns. Machine learning models, on the other hand, are generally easier to understand as well as more effective in terms of computation. By combining both approaches, the suggested methodology enables a thorough evaluation of how various fusion strategies affect XSS detection accuracy. The first suggested structure uses a statistical and contextual feature fusion technique to transform text payloads into numerous representations. hand-crafted pattern-based indications, such as the presence of <script> tags or suspicious protocols, are merged with character- and word-level TF-IDF vectors. Contextual embeddings are retrieved using LSTM or GRU networks. Finally, all these extracted features will be combined together. Following the fusion phase, the combined feature space is passed through dense layers with regularization before reaching the final classification phase with sigmoid function. This end-to-end design enables the model to learn from statistical, structural, and contextual perspectives simultaneously, as shown in Figure 1.

The second proposed model follows a late fusion strategy integrated with classical machine learning classifiers. Here, the methodology is by using LSTM or GRU models, which act as feature extractors to generate deep sequential representations of the payloads. These representations are then concatenated with Character- and word-level TF-IDF vectors are combined with hand-designed pattern-based indicators (optionally reduced through PCA), and the combined vectors are fed into conventional classifiers such as Support Vector Machines (SVM), Random Forests (RF), Logistic Regression (LR), and Multi-Layer Perceptron (MLP). The interpretability and adaptability of the proposed framework are the main goals of this design. The usage of the same fused feature space by multiple learning paradigms can be directly observed, as shown in Figure 2.

The two proposed models provide a more comprehensive and accurate experimental strategy. They not only present two different methodologies, but also help identify the best-performing model and illustrate how feature combinations and model selection interact to enhance XSS detection systems.

3.2 Dataset Description

The experiments used a publicly accessible labeled dataset of 13,686 XSS payloads created by Syed Saqlain Hussain and uploaded on Kaggle [27]. Each raw payload text is classified by a binary label, malicious (1) or safe (0), and is included in every record. No additional filtering or preprocessing was done, except to confirm the labeling accuracy and preserve consistent text encoding.

Along with valid inputs that reflect common, benign usage, the collection includes a broad range of injection techniques, including script tags, event handlers, obfuscation, and encoded payloads. For the models to learn how to identify assaults that avoid traditional filters, obfuscated samples were purposefully kept in order to capture the type of distorted payloads frequently observed in practice.

The data were separated using a stratified 80/20 split, maintaining the same class balance in both subsets, to guarantee a fair evaluation. 10% more of the training set was reserved for validation while the deep-learning parameters were being adjusted. 2,738 testing, 1,095 validation, and 9,853 training samples were generated as a result. Both fusion frameworks were subjected to the same partitioning technique in

Table 1: Related works details: techniques, results, methods, and dataset description.

Ref.	Author (Year)	Techniques	Accuracy %	Precision %	Recal l %	F1-score %	Methods	Dataset details
[19]	Muralitharan Krishnan et al. (2024)	Hybrid Stacking (NB, SVM, k-NN + RNN, CNN, LSTM)	~99.50	~99.50	~99.5 0	~99.50	Stacking combines ML and DL to boost performance	GitHub + Kaggle datasets (3,018; 4,197; 25,828; 51,656)
[20]	Wang et al. (2024) (DoubleR)	PU Learning + Bagging GBDT (Requests & Responses)	~94.00	86.00	094.0 0	-	Distinguishin g attack landscapes through request and response analysis	Requests: 5,872 malicious; 7,343 benign. Responses: ~15,000 samples (from multiple CMS)
[21]	Tadhani et al. (2024)	CNN + LSTM + Word2Vec	99.20–99.80	-	-	-	A hybrid model for SQLi/XSS detection	CSIC-2010 (not explicitly numbered), Kaggle (not explicitly numbered), Burp-generated testbed (custom)
[22]	Liu et al. (2025)	CNN + LSTM + Transformer + AdaBoost (parallel)	99.73 / 99.52	-	-	-	Combining multiple branch outputs with AdaBoost	Dataset1 (6,313 benign; 7,373 malicious), Dataset2 (~34,500 benign; ~30,110 malicious)
[23]	Zhou et al. (2025)	LLM (GPT-4) + Prompt Optimizer + Program Dependency Graphs	93.00	-	-	92.66	Using LLM to understand JS code against Stored XSS	BeEF (~610 attack samples), JS150k (~305 benign samples)
[25]	Xu et al. (2025)	Dueling Double DQN (RL) + Payload Mutation	-	-	-	95.40	Interactive reinforcemen t learning to detect XSS vulnerabilitie s	Evaluated on 10 web apps (OWASP Benchmark, DVWA, WAVSEP) covering ~347 vulnerabilities
[24]	Li et al. (2025)	CNN + BiLSTM + Multi-Head Attention	99.38	99.36	99.36	99.37	Local feature extraction + long-term tracking + Attention	XSSed-DMOZ (~64,800), Merwani-XSS (~42,600)
[26]	Abuda (2025)	NLP (TF-IDF, Lemmatization) + DNN + RL (Q-learning)	Train 88.9 \ validation 88.1	-	-	-	A hybrid framework combining NLP + RL	Kaggle (~1,300 samples)
	Our Proposed Model	GRU+TF-IDF with MLP classifier / GRU+TF-IDF	99.85 / 99.90	99.80 / 99.91	99.86 / 99.91	99.83 / 99.91	Late fusion / Early fusion	Dataset (6,313 benign; 7,373 malicious)

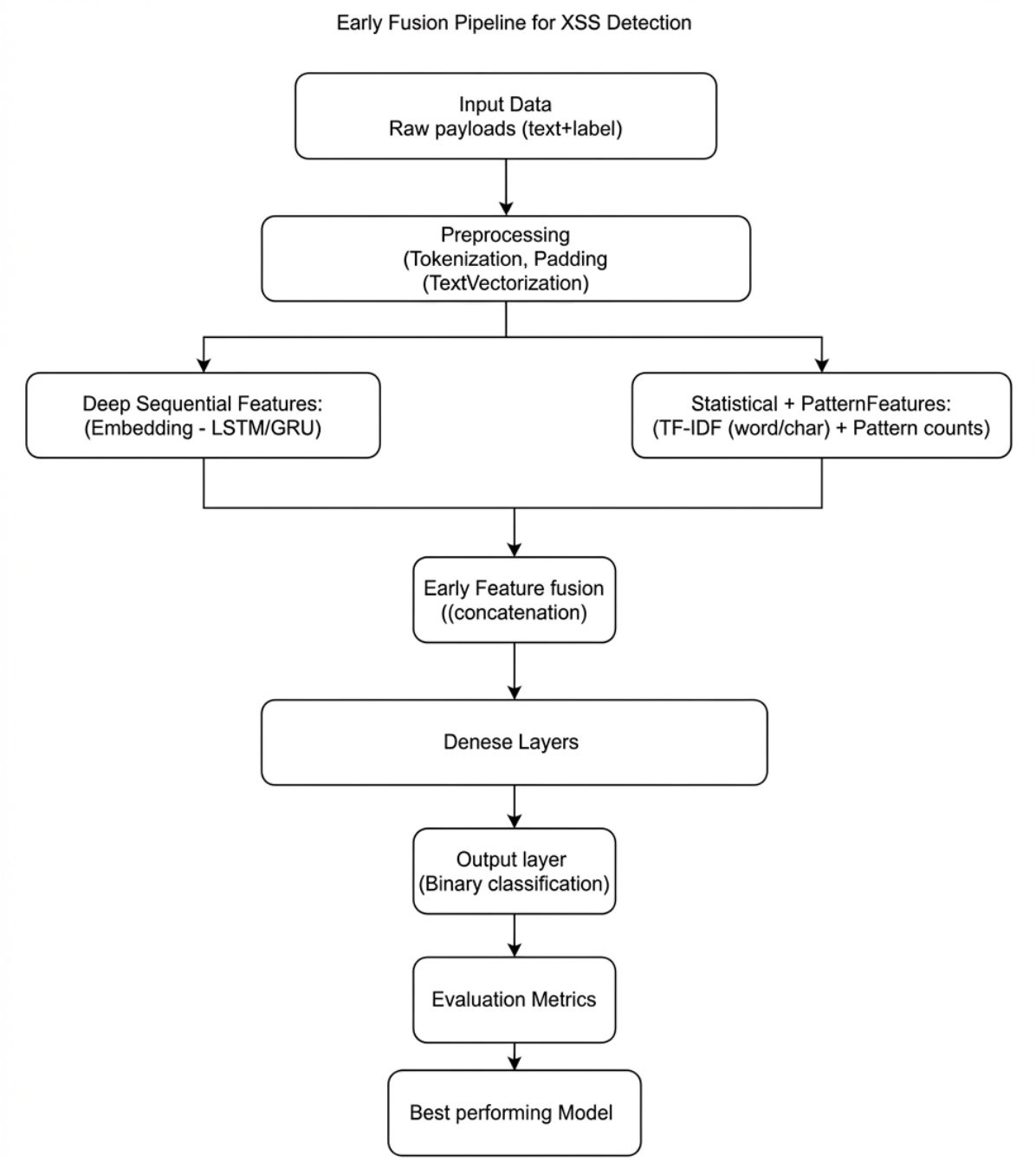

Figure 1: Early fusion pipeline.

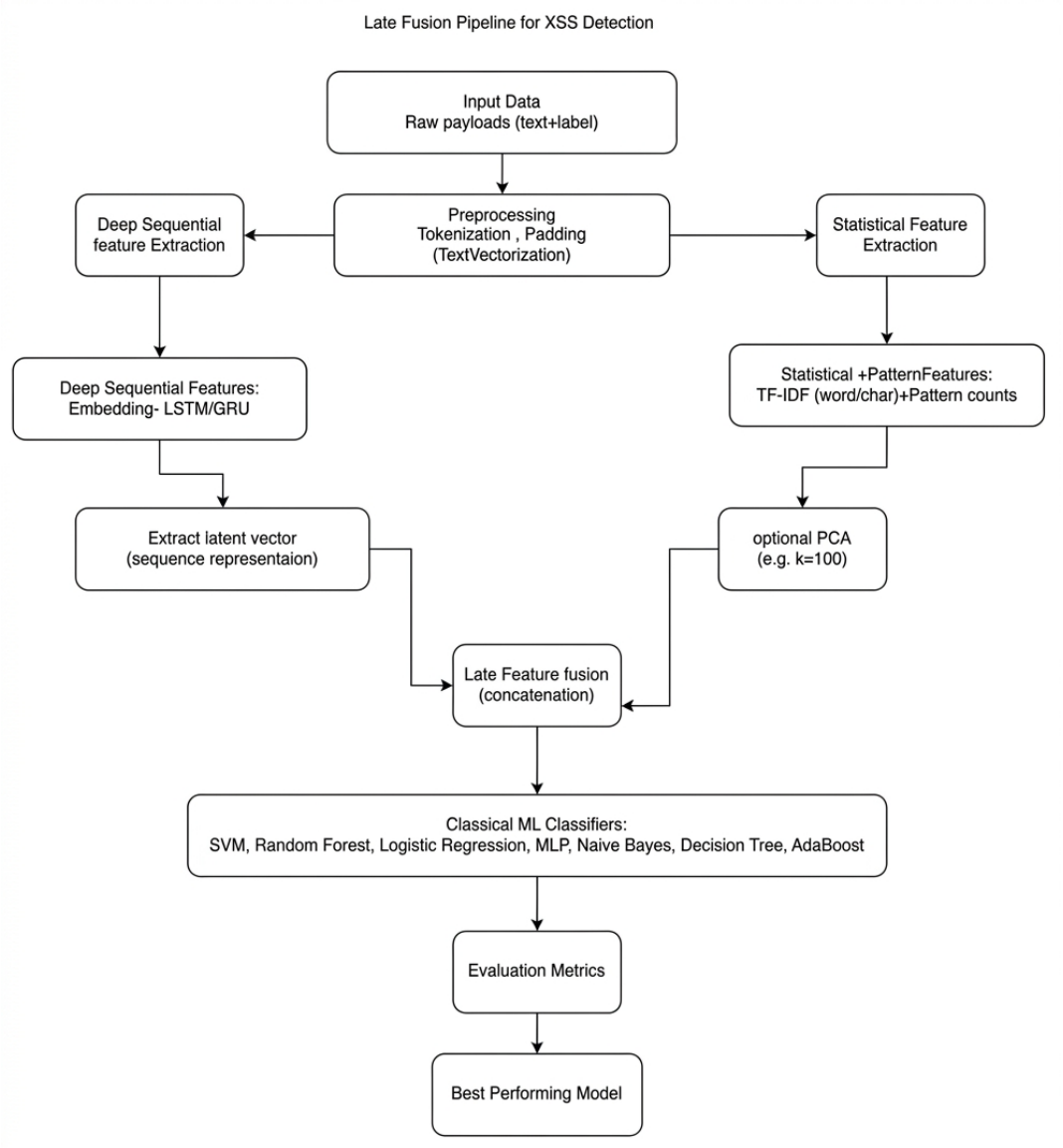

Figure 2: Late fusion pipeline.

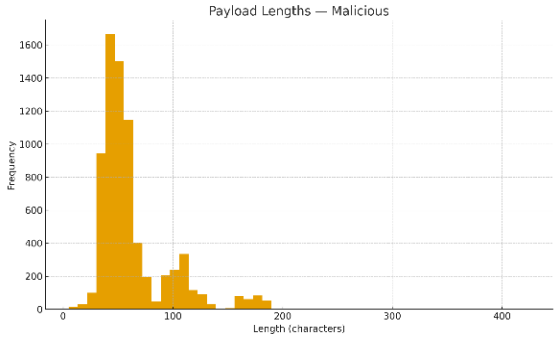

Figure 3: Payloads length of malicious samples.

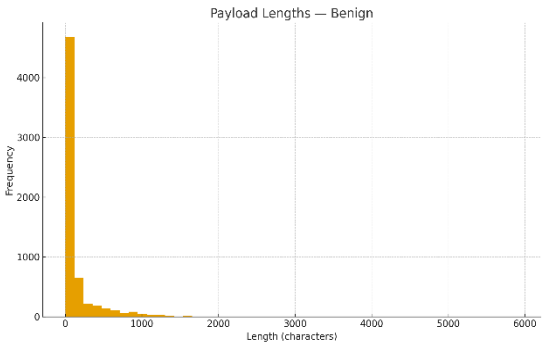

Figure 4: Payloads length of benign samples.

3.3 Data Preprocessing

The XSS attack payloads were converted into three complementary feature representations in order to prepare the dataset for training. These representations are meant to encompass the semantic, syntactic, and contextual aspects of the texts.

1) Sequential features (for deep models). Payloads were tokenized, converted into integer sequences, and padded to a fixed length suitable for LSTM and GRU. This preserves the syntactic order of tokens and enables deep models to capture contextual attack patterns.
2) Statistical features. Word- and character-level TF–IDF vectors were generated to capture lexical frequency patterns. Character-level TF–IDF is particularly effective in modeling obfuscation and short script fragments, while word-level vectors capture longer semantic units. For Late Fusion experiments, PCA was optionally applied to reduce dimensionality.
3) Pattern-based features. Handcrafted indicators were extracted, such as the presence of <script> tags, suspicious characters (<, >, "), or JavaScript protocols. These directly highlight common structural cues of malicious payloads.

All payloads were loaded in UTF-8 encoding and empty entries were removed. No HTML/JavaScript or URL decoding was applied; payloads were kept in their raw form to preserve obfuscations and structural markers essential for learning. Special symbols (e.g., <, >, ", /) were retained, and text lowercasing was applied only within the pattern-feature extractor for consistent counting.

Tokenization was implemented using Keras' Tokenizer and sequences were padded to a fixed length of 100 tokens prior to modeling, Both the early- and late-fusion experiments used a fixed random state =42 to ensure consistent and reproducible data splits across all runs.

For the Early Fusion technique, all three feature sets were concatenated within a single deep architecture, producing an end-to-end representation for classification. In contrast, the Late Fusion technique employed LSTM/GRU only as feature extractors; their latent representations were concatenated with TF–IDF/PCA vectors and passed to classical classifiers (e.g., SVM, RF, LR, MLP).

No additional cleaning such as stop-word removal or deduplication was applied, as XSS payloads often rely on distorted or incomplete tokens that must be preserved.

3.4 Late Fusion with Classical Machine Learning

The second framework adopts a late fusion strategy, where deep models serve as feature extractors rather than end-to-end classifiers. Each payload was first transformed into token sequences and passed through pretrained LSTM or GRU models to generate latent sequential embeddings (seq_repr). These high-level representations were concatenated with TF–IDF vectors, optionally dimensionality-reduced using Principal Component Analysis (PCA), into a hybrid feature space.

Traditional machine learning classifiers, including Support Vector Machines (SVM), Random Forests (RF), Logistic Regression (LR), Decision Trees (DT), AdaBoost, and Multi-Layer Perceptron (MLP), were applied to the generated vectors. This architecture allows for the evaluation of the parallel integration of statistical, engineered features and features extracted based on deep representations through traditional machine learning methods. The late fusion approach is more interpretability- and flexibility-oriented than the early fusion method, which concatenates every feature in a deep neural network. Since feature extraction and classification are decoupled, one can examine each classifier's contribution and balance accuracy, computational efficiency, and generalization capability.

The strategy of late fusion was based on integrating the statistical TF-IDF features (max features = 1000) with deep contextual embeddings from LSTM or GRU models (128 hidden units, dropout = 0.3). The embeddings were taken from the final hidden layer (32-dimensional) and concatenated with TF–IDF vectors. PCA was optionally applied on the TF–IDF features to reduce dimensionality before training traditional classifiers, including SVM (RBF kernel), Random Forest (100 trees), Logistic Regression, and MLP (100–50 hidden units, ReLU activation). This pipeline allowed efficient integration of deep and statistical representations within a unified decision stage.

3.5 Early Fusion with Pattern Features

The proposed model follows an early-fusion strategy in which different types of feature representations are combined within a single deep-learning framework. A padded token sequence, a word- and character-level TF-IDF vector, and a collection of handcrafted indicators that identify particular script patterns are all encoded simultaneously in each

payload. At the feature level, these disparate aspects are combined to create a joint representation that captures the data's syntactic, statistical, and structural characteristics.

Following that, the fused vectors are sent via recurrent layers (LSTM or GRU), which are able to recognize contextual patterns and sequential dependencies among the payloads. The learned representations are refined by fully linked dense layers after dropout layers are added to avoid overfitting. To differentiate between benign and malicious inputs, the output layer uses binary classification, which is triggered by a sigmoid function.

The model may use both automatically learned deep features and explicit handmade inputs in a single, end-to-end learning process due to this approach. The model is better equipped to identify obfuscated or unconventional attack payloads that could otherwise evade conventional single-view detection systems by exposing the network to several complementary feature perspectives from the outset.

3.6 Training and Hyperparameter Settings

The standard setup for binary text classification, which includes the Adam optimizer and binary cross-entropy loss, was used to train the deep learning models. For final classification, each model employed a sigmoid output after dense layers activated with ReLU and a single LSTM or GRU layer with 128 hidden units. Training was done with an embedding dimension of 100 and a batch size of 64 over 50 epochs. Dropout layers were added at a rate of 0.3 to lessen overfitting and enhance the model's generalizability.

PCA (100 components) was an optional method used to reduce the dimensionality of the TF-IDF vectors in the late-fusion framework. Following this stage, several conventional classifiers (SVM, Random Forest, Logistic Regression, and MLP) were trained using the combined deep and statistical features, each with its default hyperparameter values.

4 RESULTS

4.1 Results of the Late Fusion Approach

When compared to employing a single feature space, the classifiers' capacity to discriminate between benign and malicious payloads was significantly enhanced by integrating the LSTM and GRU embeddings with TF-IDF features.

With a 99.82% accuracy, a 99.99% ROC-AUC, and a 99.82% F1-score, the Multi-Layer Perceptron (MLP) produced the high performance results in the LSTM + TF–IDF approach. Likewise, the MLP has once again shown superior performance in the GRU + TF–IDF setup, achieving 99.85% accuracy, 99.99% ROC-AUC, and a 99.86% F1-score. The capacity of other classifiers, including Random Forest and Decision Tree, to capture the non-linear relations within the fused feature space was demonstrated by their strong and reliable results (F1-scores above 99.5%).

However, the use of Gaussian Naïve Bayes produced considerably fewer effective results (89.15% accuracy, 90.82% F1-score), indicating that the sophisticated sequential–statistical relationships present in XSS payloads cannot be captured by simple statistical assumptions only. These results are shown as in Tables 2 - 5, further highlighted by the visually appealing analysis in Figures 5 - 10.

When using sequential features alone, the GRU-only and LSTM-only embeddings, as shown in Figures 5 and 6, mainly overlap regions, indicating insufficient class separation. When TF-IDF is included, the distinctions between malicious and benign samples become much more obvious, as seen in Figures 7 and 8. Adding PCA to the fusion step reduces computational cost and enhances visual distinction by eliminating redundancy but results in a slight decrease in classification accuracy, as shown in Figures 9 and 10.

Overall, these findings demonstrate that the late-fusion approach provides a useful and reliable way to identify XSS. The approach offers dependable, high-accuracy performance by fusing statistical features with contextual sequence information. The good outcomes of Random Forest and Logistic Regression demonstrate their stability and emphasize the necessity of careful classifier selection in order to fully benefit from feature fusion.

Table 2: Performance comparison of classical classifiers under Late Fusion with LSTM + TF-IDF features on the testing dataset.

Combination	Classifier	Accuracy	ROC-AUC	Precision	Recall	F1-Score
LSTM + TFIDF	RandomForest	99.60	99.99	1	99.25	99.63
LSTM + TFIDF	AdaBoost	99.53	99.98	99.93	99.19	99.56
LSTM + TFIDF	DecisionTree	99.56	99.57	99.73	99.46	99.59
LSTM + TFIDF	LogisticRegression	99.78	99.99	1	99.59	99.8
LSTM + TFIDF	GaussianNB	89.15	88.30	83.43	99.66	90.82
LSTM + TFIDF	SVM	99.74	99.96	99.86	99.66	99.76
LSTM + TFIDF	MLPClassifier	99.82	1	99.86	99.80	99.83

Table 3: Performance comparison of classical classifiers under Late Fusion with GRU + TF-IDF features on the testing dataset.

Combination	Classifier	Accuracy	ROC-AUC	Precision	Recall	F1-Score
GRU + TFIDF	RandomForest	99.67	99.99	1	99.39	99.69
GRU + TFIDF	AdaBoost	99.53	99.99	1	99.12	99.56
GRU + TFIDF	DecisionTree	99.56	99.58	99.8	99.39	99.59
GRU + TFIDF	LogisticRegression	99.78	1	1	99.59	99.8
GRU + TFIDF	GaussianNB	89.15	88.3	83.43	99.66	90.82
GRU + TFIDF	SVM	99.74	99.98	99.86	99.66	99.76
GRU + TFIDF	MLPClassifier	99.85	1	99.93	99.80	99.86

Table 4: Performance comparison of classical classifiers under Late Fusion with PCA-LSTM + TF-IDF features on the testing dataset.

Combination	Classifier	Accuracy	ROC-AUC	Precision	Recall	F1-Score
PCA-LSTM + TFIDF	RandomForest	99.63	1	1	99.32	99.66
PCA-LSTM + TFIDF	AdaBoost	99.45	99.97	99.93	99.05	99.49
PCA-LSTM + TFIDF	DecisionTree	99.60	99.62	99.93	99.32	99.63
PCA-LSTM + TFIDF	LogisticRegression	99.78	99.99	1	99.59	99.80
PCA-LSTM + TFIDF	GaussianNB	89.15	88.3	83.43	99.66	90.82
PCA-LSTM + TFIDF	SVM	99.74	99.96	99.86	99.66	99.76
PCA-LSTM + TFIDF	MLPClassifier	99.78	1	99.80	99.80	99.80

Table 5: Performance comparison of classical classifiers under Late Fusion with PCA-GRU + TF-IDF features on the testing dataset.

Combination	Classifier	Accuracy	ROC-AUC	Precision	Recall	F1-Score
PCA-GRU + TFIDF	RandomForest	99.67	1	1	99.39	99.69
PCA-GRU + TFIDF	AdaBoost	99.53	99.98	1	99.12	99.56
PCA-GRU + TFIDF	DecisionTree	99.53	99.55	99.86	99.25	99.56
PCA-GRU + TFIDF	LogisticRegression	99.78	1	1	99.59	99.80
PCA-GRU + TFIDF	GaussianNB	89.15	88.3	83.43	99.66	90.82
PCA-GRU + TFIDF	SVM	99.74	99.98	99.86	99.66	99.76
PCA-GRU + TFIDF	MLPClassifier	99.78	1	99.80	99.80	99.80

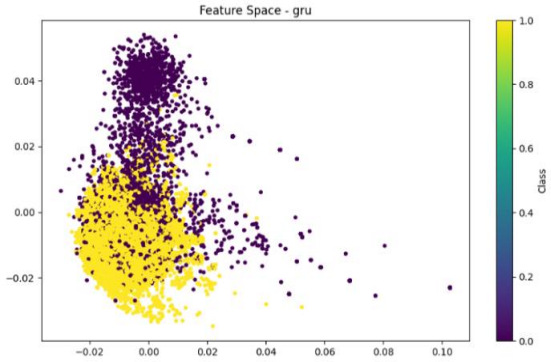

Figure 5: Feature-space visualization using GRU-based embeddings. Yellow points correspond to malicious (XSS) payloads, while dark-purple points indicate benign inputs.

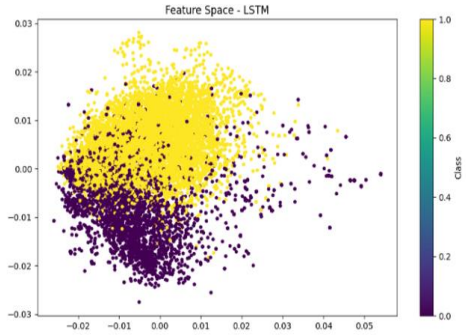

Figure 6: Feature-space visualization using LSTM-based embeddings. Yellow points correspond to malicious (XSS) payloads, while dark-purple points indicate benign inputs.

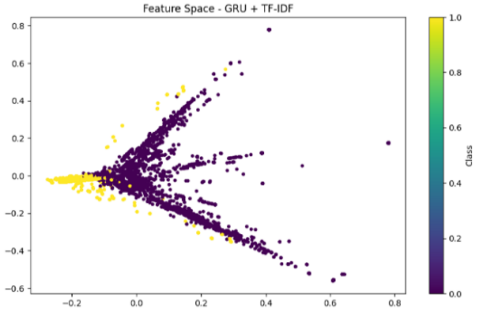

Figure 7: Feature-space visualization using GRU + TF-IDF based embeddings. Yellow points correspond to malicious (XSS) payloads, while dark-purple points indicate benign inputs.

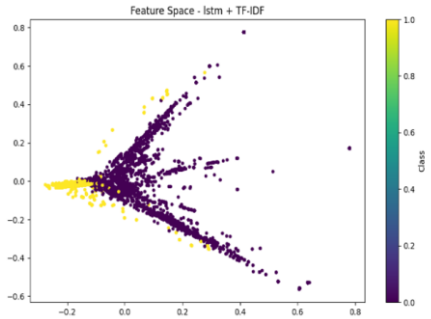

Figure 8: Feature-space visualization using LSTM + TF-IDF based embeddings. Yellow points correspond to malicious (XSS) payloads, while dark-purple points indicate benign inputs.

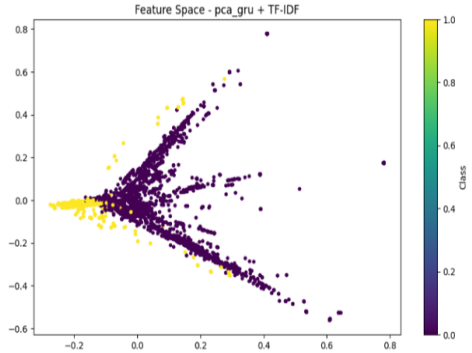

Figure 9: Feature-space visualization using PCA – GRU + TF-IDF based embeddings. Yellow points correspond to malicious (XSS) payloads, while dark-purple points indicate benign inputs.

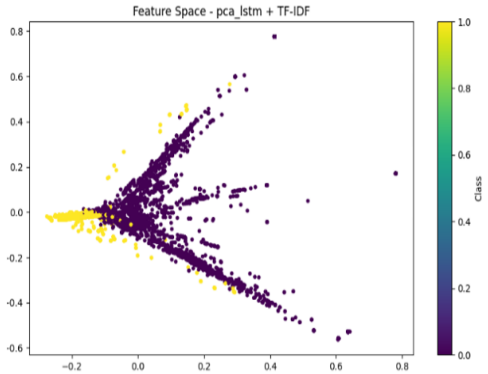

Figure 10: Feature-space visualization using PCA – LSTM + TF-IDF based embeddings. Yellow points correspond to malicious (XSS) payloads, while dark-purple points indicate benign inputs.

Table 6: Results that achieved based on early fusion models with the testing dataset.

Model	Accuracy	ROC-AUC	F1-Score	Precision	Recall
LSTM+TFIDF	99.90	99.90	99.91	99.91	99.91
GRU+TFIDF	99.87	99.87	99.89	99.86	99.91
PCA+GRU+TFIDF	98.97	98.99	99.05	99.36	98.73
PCA+LSTM+TFIDF	98.65	98.62	98.76	98.60	98.92

4.2 Results of the Early Fusion Approach

The model developed based on early fusion achieved high performance accuracy, exemplifying the strength of combining sequential, statistical, and pattern-based (hand-crafted) features in the unified deep learning network. Table 6 demonstrates the results of four early fusion models, such as (GRU+TFIDF, LSTM+TFIDF, PCA+GRU+TFIDF, and PCA+LSTM+TFIDF). These models were tested and evaluated by the most important metrics of the detection performance, like accuracy, precision, recall, F1-score, and ROC-AUC. Generally, the GRU and LSTM models with TF-IDF performed reasonably well performance in detection. The model that is based on GRU+TFIDF achieved good results: an accuracy score of 99.87% and an F1-score of 99.89% and presents a rational ability to efficiently detect malicious payloads. Similarly, the other model, LSTM+TFIDF, was evidenced by the enhanced overall performance: an accuracy score of 99.90% and an F1-score of 99.91%.

It is important to note that the GRU and LSTM models' overall performance accuracy scores, which were 98.65% and 98.97%, respectively, slightly decreased once Principal Component Analysis (PCA) was added. Despite this minor reduction, both models maintained high recall and precision rates of over 98.6%. This method illustrates that, while Principal Component Analysis (PCA) efficiently decreases computational costs and feature dimensions, it also eliminates some discriminative indicators that aid models in detecting highly obfuscated attack payloads.

These results support the reliability of the proposed architecture of the early fusion model for XSS detection. By integrating TF-IDF, statistical representations, and feature engineering into a single deep learning architecture, this strategy yields rational and reliable results. Its performance is comparable to modern, widely used systems like CNN, BiLSTM-Attention, and transformer-based models, although it has a lower computational complexity and is simpler to comprehend.

5 DISCUSSION

This work presents a methodology and functional architecture for two different approaches with a strategy that leads to the identification and detection of XSS attacks. The presented models achieved outstanding findings for major assessment metrics rates; the most prominent of these was the early fusion model, which uses TF-IDF and LSTM representations. The framework demonstrates that it can detect complex and ambiguous payloads. Since the payloads were not decrypted or normalized and left in their original form, the model was independently trained to recognize XSS-deformed attacks, strengthening its defenses against sophisticated forms of attacks. Furthermore, the late fusion technique, which included an MLP classifier, GRU, and TF-IDF, outperformed the other classifiers examined in this method. In contrast, principal component analysis (PCA) gave a more adaptable and understandable approach. Because numerous sensitive and critical feature indicators were deleted, detection sensitivity was greatly lowered; however, this limitation did not prevent principal components analysis from benefiting from lower computational costs.

Overall, the study provides clear and practical results with a comparison between detection accuracy and model interpretability. The effectiveness of the fusion techniques for real-world XSS detection tasks is demonstrated through the proposed methods that outperformed previous research and studies, based on the main performance accuracy measures, especially when using the same dataset.

6 REPRODUCIBILITY AND ETHICAL

For the purpose of research verification, the whole implementation (including preprocessing procedures and model configurations) will be made available upon reasonable request. The study's dataset comes from an open-data license-issued Kaggle repository that is accessible to the public. Its use conforms with

the dataset's stated terms and conditions and doesn't contain any sensitive or personal information.

7 CONCLUSIONS

This study proposed and evaluated two feature fusion architectures for detecting Cross-Site Scripting (XSS) attacks. The early fusion model combined sequential embeddings from LSTM networks with TF-IDF representations and hand-crafted pattern features within a unified deep architecture, enabling the simultaneous capture of structural and contextual dependencies, resulting in high performance (99.90% accuracy and 99.91% F1-score). Its effectiveness was particularly evident when handling highly obfuscated payloads, which were processed in their original form during preprocessing.

The late fusion employed a more uniform and comprehensible pipeline in which DL models served as feature extractors and conventional ML methods were used to classify the combined representations. Although its performance was slightly lower than the early fusion, the GRU+TF-IDF+MLP configuration achieved good results (99.85% accuracy and 99.86% F1-score). This shows that separating feature extraction from classification enables flexibility and adaptation.

In general, the findings demonstrate that integrating contextual, statistical, and pattern-based representations outperforms single-feature or single-model methods. As proposed models showed high generalizability regarding the dataset, future developments may include resilience against more complex obfuscated payloads, and the inclusion of attention-based structures or graph-based architectures, to increase usefulness to detection.

REFERENCES

[1] M. S. Chughtai, I. Bibi, S. Karim, S. W. A. Shah, A. A. Laghari, and A. A. Khan, “Deep learning trends and future perspectives of web security and vulnerabilities,” J. High Speed Netw., vol. 30, no. 1, pp. 115–146, 2024.

[2] I. J. Mohmmed, B. T. Al-Nuaimi, and D. I. S. Bakr, “Machine learning prediction models applied to weather forecasting: A survey,” Iraqi J. Sci. Eng. Res., vol. 1, no. 2, pp. 80–85, 2023.

[3] A. Hannousse, S. Yahiouche, and M. C. Nait-Hamoud, “Twenty-two years since revealing cross-site scripting attacks: A systematic mapping and a comprehensive survey,” Comput. Sci. Rev., vol. 52, pp. 100634, 2024.

[4] W. S. Ahmed, Z. T. M. Al-Ta’i, T. Abegaz, and G. S. Mahmood, “Digital forensics architecture for real-time automated evidence collection and centralization: Leveraging security lake and modern data architecture,” J. Intell. Syst., vol. 33, no. 1, pp. 20240109, 2024.

[5] T. Neumann, “Cybersecurity in maritime industry,” TransNav: Int. J. Mar. Navig. Saf. Sea Transp., vol. 18, 2024.

[6] A. S. Hussainy, M. A. Khalifa, A. Elsayed, A. Hussien, and M. A. Razek, “Deep learning toward preventing web attacks,” in Proc. IEEE Conf. on Deep Learning Toward Preventing Web Attacks, 2022, pp. 280–285.

[7] J. Kaur, U. Garg, and G. Bathla, “Detection of cross-site scripting (XSS) attacks using machine learning techniques: A review,” Artif. Intell. Rev., vol. 56, no. 11, pp. 12725–12769, 2023.

[8] S. Alazmi and D. C. De Leon, “A systematic literature review on the characteristics and effectiveness of web application vulnerability scanners,” IEEE Access, vol. 10, pp. 33200–33219, 2022.

[9] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, “A comprehensive review of cybersecurity vulnerabilities, threats, attacks, and solutions,” Electronics, vol. 12, no. 6, pp. 1333, 2023.

[10] X. Luo, J. Li, W. Wang, Y. Gao, and W. Zhao, “Towards improving detection performance for malware with a correntropy-based deep learning method,” Digit. Commun. Netw., vol. 7, no. 4, pp. 570–579, 2021.

[11] 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, pp. 2212, 2022.

[12] W. Melicher, C. Fung, L. Bauer, and L. Jia, “Towards a lightweight, hybrid approach for detecting DOM XSS vulnerabilities with machine learning,” in Proc. Int. Conf. on Web Security, 2021, pp. 2684–2695.

[13] R. Alhamyani and M. Alshammari, “Machine learning-driven detection of cross-site scripting attacks,” Information, vol. 15, no. 7, pp. 420, 2024.

[14] G. Rodríguez-Galán and J. Torres, “Personal data filtering: A systematic literature review comparing the effectiveness of XSS attacks in web applications vs cookie stealing,” Ann. Telecommun., vol. 79, no. 11, pp. 763–802, 2024.

[15] D. F. Somé, “MatriXSSed: A new taxonomy for XSS in the modern web,” in Proc. Int. Conf. on Web Applications Security, 2025, pp. 4662–4672.

[16] M. Paramesha, N. Rane, and J. Rane, “Artificial intelligence, machine learning, and deep learning for cybersecurity solutions: A review of emerging technologies and applications,” Mach. Learn. Deep Learn. Cybersecurity Solut., pp. 1–20, Jun. 2024.

[17] M. Liu, B. Zhang, W. Chen, and X. Zhang, “A survey of exploitation and detection methods of XSS vulnerabilities,” IEEE Access, vol. 7, pp. 182004–182016, 2019.

[18] 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, pp. 3483, 2025.

[19] M. Krishnan, Y. Lim, S. Perumal, and G. Palanisamy, "Detection and defending the XSS attack using novel hybrid stacking ensemble learning-based DNN approach," Digit. Commun. Netw., vol. 10, no. 3, pp. 716–727, Jun. 2024.

[20] W. Wang, P. Yi, and H. Xu, "DoubleR: Effective XSS attacking reality detection," Comput. Netw., vol. 251, pp. 110567, 2024.

[21] J. R. Tadhani, V. Vekariya, V. Sorathiya, S. Alshathri, and W. El-Shafai, "Securing web applications against XSS and SQLi attacks using a novel deep learning approach," Sci. Rep., vol. 14, no. 1, pp. 1803, 2024.

[22] Y. Liu, L. Wang, and Y. Dai, "XSS attack detection with deep learning and AdaBoost," Update, vol. 100, pp. 4, 2025.

[23] Y. Zhou, E. Wang, W. Yang, W. Ge, S. Yang, Y. Zhang, W. Qu, and W. Xie, "XSShield: Defending against stored XSS attacks using LLM-based semantic understanding," Appl. Sci., vol. 15, no. 6, pp. 3348, 2025.

[24] Z. Li, F. Liu, Z. Gu, and Y. Liu, "XSS attack detection method based on CNN-BiLSTM-Attention," Appl. Sci., vol. 15, no. 16, pp. 8924, 2025.

[25] K. Xu, H. He, Y. Zhao, Y. Jia, P. Shi, and B. Zhang, "An adaptive XSS vulnerability detection method based on hierarchical multi-objective reward-enhanced dueling double deep Q-network," Comput. Netw., pp. 111595, 2025.

[26] C. J. P. Abuda, "Hybrid detection framework using natural language processing (NLP) and reinforcement learning (RL) for cross-site scripting (XSS) attacks," Int. J. Adv. Comput. Sci. Appl., vol. 16, no. 6, 2025.

[27] S. S. H. Hussain, "Cross-site scripting (XSS) dataset for deep learning," Kaggle Dataset, 2020. [Online]. Available: https://www.kaggle.com/datasets/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning. Accessed: Apr. 12, 2025.