International Journal Software Engineering and Computer Science (IJSECS)
6 (1), 2026, 391-403
Published Online April 2026 in IJSECS (http://www.journal.lembagakita.org/index.php/ijsecs)
P-ISSN: 2776-4869, E-ISSN: 2776-3242. DOI: https://doi.org/10.35870/ijsecs.v6i1.7193.

RESEARCH ARTICLE Open Access

Web Attack Detection for SQLi and XSS Using Ensemble Learning Based on Character-Level N-Gram Features

Yaya Suharya [1], **Mohammad Bayu Anggara** [2*].

[1,2*] Department of Informatics Engineering, Universitas Bale Bandung, Bandung Regency, West Java Province, Indonesia.

**Corresponding author*: mohammadbayuanggara@gmail.com.

Received: April 12, 2026; Accepted: April 25, 2026; Published: April 30, 2026.

Abstract: SQL Injection (SQLi) and Cross-Site Scripting (XSS) remain severe threats to web application security, particularly as attackers employ increasingly sophisticated obfuscation techniques to bypass conventional detection systems. This research constructs a machine learning framework using ensemble learning — specifically combining Random Forest and XGBoost — integrated with character-level n-gram feature extraction. The methodology involved rigorous data curation of a large-scale dataset, refining 156,636 raw samples into 151,783 unique entries to ensure high-quality training data. By extracting 10,000 character-level n-gram features, the model captures the intricate structural patterns of complex and obfuscated payloads. Experimental results show consistent and measurable performance: the proposed ensemble model achieved an overall accuracy of 99.67%. Stability was confirmed through a 5-fold cross-validation process, yielding a mean accuracy of 99.64% and a standard deviation of 0.0003. These findings are reinforced by ROC AUC scores of 1.0000 for XSS and 0.9999 for SQLi, indicating near-perfect discriminative capability. The combination of character-level representation and ensemble learning produces a precise and resilient solution for safeguarding modern web environments against dynamic and evolving cyber threats.

Keywords: Character-level N-gram; Ensemble Learning; SQL Injection; Web Security; XSS.

1. Introduction

The rapid proliferation of web-based applications has directly amplified the frequency of digital transactions and information exchange in cyberspace — and with it, the attack surface available to malicious actors (Ahmed & Uddin, 2020). The consequences are far from abstract: government agencies, private enterprises, and individual users alike have sustained measurable damage from web-based intrusions, a pattern that shows no sign of abating despite sustained global mitigation efforts such as the OWASP Top 10 (Feng *et al.*, 2024; Patil & Bansode, 2024). Firewall mechanisms, while useful, carry well-documented structural limitations that allow adversaries to penetrate web servers and network assets with relative ease (Kshirsagar & Kumar, 2020). Compounding this, a persistent deficiency in security literacy among developers continues to jeopardize application integrity — amplifying the probability of data manipulation and leakage as the number of human actors interacting within these systems grows (Riadi *et al.*, 2020).

Among the attack categories that consistently appear at the top of vulnerability rankings, SQL Injection (SQLi) and Cross-Site Scripting (XSS) are particularly consequential. Both exploit weaknesses in web

application source code to gain unauthorized access, manipulate data, or hijack user sessions (Bakır, 2025; Li *et al.*, 2025; Tadhani *et al.*, 2024). According to OWASP classifications, SQL Injection enables attackers to infiltrate database systems and extract confidential information illegally — and the prevalence figures are striking: SQLi is present in 43% of applications, while XSS reaches 61% (Bakır, 2025; Odeh & Taleb, 2024). Parsing techniques and pattern blacklists have historically served as the primary countermeasures (Sonoda, 2011), yet these approaches are structurally ill-equipped to handle the obfuscation strategies that modern attackers routinely deploy.

Signature-based Web Application Firewalls (WAFs) provide a baseline defensive layer, but their fundamental weakness is well established — they fail against novel attack variants and polymorphic payloads capable of altering their own structure to evade detection (Bakır, 2025; Işiker, 2021; Panadiya *et al.*, 2024). The problem runs deeper than WAF design, however. Word-level and signature-based detection methods share a common blind spot: they cannot reliably identify attacks whose structures have been granularly modified at the character level (Ren *et al.*, 2018). Machine learning and deep learning approaches offer a more adaptive alternative, yet their computational complexity frequently impedes practical deployment, and overconfident predictions in these models introduce a distinct category of risk (Le *et al.*, 2024; Odeh & Taleb, 2024; Sornsuwit & Jaiyen, 2019). Neither paradigm, taken alone, resolves the detection problem satisfactorily.

Character-level n-gram models combined with ensemble learning represent a more tractable path forward. Phuong *et al.* (2020) demonstrated their effectiveness in homograph attack classification, and the underlying logic transfers directly to SQLi and XSS contexts: by decomposing payloads into their smallest structural fragments, the approach captures anomalies that word-level tokenization misses entirely. Ensemble methods further strengthen this by combining multiple base learners — reducing both bias and variance beyond what any single model achieves independently (Lower & Zhan, 2020). Habibi & Surantha (2020) confirmed that n-gram integration significantly improves XSS detection, while Le *et al.* (2024) identified a specific limitation of XGBoost when applied to SQL Injection in isolation, pointing toward the value of strategic model combination. The UniEmbed framework by Bakır (2025), which fuses sentence-level (USE), word-level (Word2Vec), and character n-gram (FastText) representations, further validates that character-level features carry substantial discriminative weight across multiple classifiers.

The gap that remains is specific — no existing framework simultaneously achieves high accuracy against obfuscated payloads, maintains statistically validated stability across data partitions, and avoids the computational overhead of deep learning architectures. This study addresses that gap directly by developing a combined Random Forest and XGBoost ensemble framework leveraging character-level N-gram features for the concurrent detection of SQL Injection and XSS attacks. The objectives are fourfold — (1) to design a character-level n-gram feature extraction pipeline capable of capturing intricate structural patterns in attack payloads; (2) to develop an ensemble model combining Random Forest and XGBoost for multi-class web attack classification; (3) to evaluate detection performance using standard classification metrics, ROC AUC analysis, and Precision-Recall curves; and (4) to validate model generalizability and statistical stability through 5-fold cross-validation and black-box testing with unseen obfuscated payloads. Each subsequent section builds on this foundation — the literature is examined critically, the methodology is laid out in full, and the results are tested against both statistical and practical benchmarks.

2. Related Work

2.1 Web Attack Detection Methods

Web attack detection has attracted substantial research attention given the persistent and evolving threat landscape. Ren *et al.* (2018) proposed a detection approach based on Bag of Words and Hidden Markov Model, showing that sequential character-level representations can effectively model attack payload behavior — though signature-based and word-level methods continue to exhibit critical limitations when confronted with novel or structurally modified attack variants. Sonoda (2011) addressed SQL Injection detection using single-character analysis, an early approach that laid the groundwork for character-granularity detection but was limited in scalability. More recently, Tadhani *et al.* (2024) developed a deep learning approach for securing web applications against XSS and SQLi attacks, achieving competitive results at the cost of higher computational complexity. Li *et al.* (2025) extended this direction further with a CNN-BiLSTM-Attention architecture for XSS detection, leveraging sequential and attention-based learning to capture contextual attack patterns; effective as it is, such architectures typically require significant computational resources and large volumes of labeled data. Bakır (2025) introduced UniEmbed, a multi-level feature fusion approach showing that the integration of character-level features significantly improves multiple classification model performances, while Patil & Bansode (2024) achieved strong performance with reduced feature dimensionality through optimized feature selection combined with machine learning-based attack detection.

2.2 Ensemble Learning in Cybersecurity

Ensemble learning has been extensively studied as a strategy to improve classification robustness and accuracy in cybersecurity domains. Lower & Zhan (2020) demonstrated that combining multiple base learners consistently outperforms individual models by reducing both bias and variance — a finding that holds across a wide range of threat classification tasks. Kshirsagar & Kumar (2020) applied this principle specifically to web attack detection, showing that dimensionality-aware ensemble approaches yield measurable improvements in detection efficiency and accuracy. Le *et al*. (2024) investigated ensemble learning and boosting models for SQL Injection detection, confirming that while GBDT and HGBDT are competitive, Random Forest and AdaBoost remain superior in overall performance; notably, XGBoost in isolation showed limitations in SQL Injection classification, suggesting the need for strategic model combinations rather than single-algorithm reliance. Kiruthika *et al*. (2024) further showed that ensemble learning frameworks are highly effective in detecting obfuscated malware — a finding directly applicable to obfuscated web attack payloads.

2.3 N-gram Feature Extraction for Attack Detection

The use of n-gram features — particularly at the character level — has proven highly effective for capturing structural anomalies in attack payloads. Habibi & Surantha (2020) demonstrated that integrating n-gram features with machine learning algorithms significantly improves XSS attack detection effectiveness, while Phuong *et al*. (2020) applied ensemble learning combined with N-gram models in homograph attack classification, recording a 1.81% improvement in accuracy and a 2.15% reduction in the false positive rate. Subba & Gupta (2021) established that TF-IDF-weighted n-gram features effectively distinguish anomalous from normal system processes, and Ahmed & Uddin (2020) confirmed that cyber attack detection benefits substantially from combining NLP-based feature representations with ensemble learning. Taken together, these studies build a consistent case: character-level granularity captures what word-level tokenization routinely misses.

2.4 Comparative Analysis and Research Positioning

The comparative analysis of prior work, summarized in Table 1, reveals that while individual classifiers and deep learning approaches achieve competitive accuracy, they present trade-offs in computational efficiency, interpretability, and robustness against obfuscated payloads. The key research gap is the absence of a framework that simultaneously addresses multi-class web attack detection (SQLi and XSS), leverages character-level granularity for obfuscation resilience, and provides statistically validated stability across data partitions.

Table 1. Comparison of Related Studies on Web Attack Detection

Study	Method	Attack Type	Key Metric	Limitation
Habibi & Surantha (2020)	ML + N-gram	XSS	Improved accuracy	Single attack type
Le *et al*. (2024)	Ensemble (GBDT, RF)	SQL Injection	Competitive accuracy	XGBoost underperforms alone
Phuong *et al*. (2020)	Ensemble + N-gram	Homograph	+1.81% accuracy	Not applied to SQLi/XSS
Bakır (2025)	UniEmbed	XSS, SQLi	Significant ML improvement	Multi-source complexity
Kiruthika *et al*. (2024)	Ensemble Learning	Obfuscated Malware	Effective detection	Not web-specific
Tadhani *et al*. (2024)	Deep Learning	XSS, SQLi	Strong results	High computational cost
Li *et al*. (2025)	CNN-BiLSTM-Attention	XSS	High accuracy	Resource-intensive
This Study	RF + XGBoost + Char N-gram	XSS, SQLi, Normal	99.67% accuracy	—

The proposed framework addresses these gaps by combining the variance-reduction strength of Random Forest with the bias-reduction capability of XGBoost, integrated with character-level n-gram features that capture the finest structural fragments of attack payloads — providing resilience against obfuscation without incurring the computational overhead of deep learning architectures.

3. Methodology

3.1 Research Stages

This research was systematically designed to construct a reliable web attack detection framework through a machine learning approach. The workflow commenced with raw dataset collection, proceeded through data cleaning, character-level feature extraction, ensemble model design, and a multi-stage statistical evaluation scheme. The complete research stages are illustrated in Figure 1.

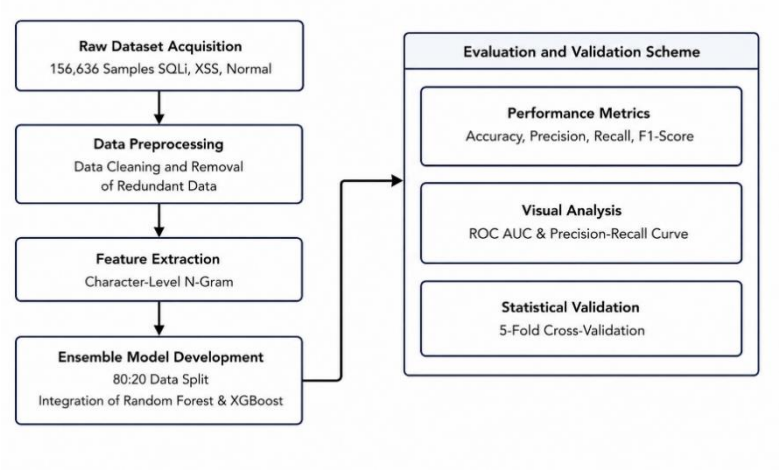

Figure 1. Research Workflow Diagram

3.2 Dataset Description

The primary dataset was sourced from the open Kaggle repository under the title "SQL Injection and XSS Mix Dataset" (https://www.kaggle.com/datasets/alextrinity/sqli-xss-dataset), specifically the file SQLInjection_XSS_MixDataset.1.0.0.csv. Prior to processing, the raw dataset comprised 156,636 samples distributed across three categories: 57,316 SQL Injection attack samples, 40,799 Cross-Site Scripting (XSS) samples, and 58,521 normal traffic samples. Since the data was collected from various traffic log sources, its characteristics were heterogeneous and contained significant redundancy and null values — necessitating a rigorous pre-processing procedure before any feature extraction could proceed reliably.

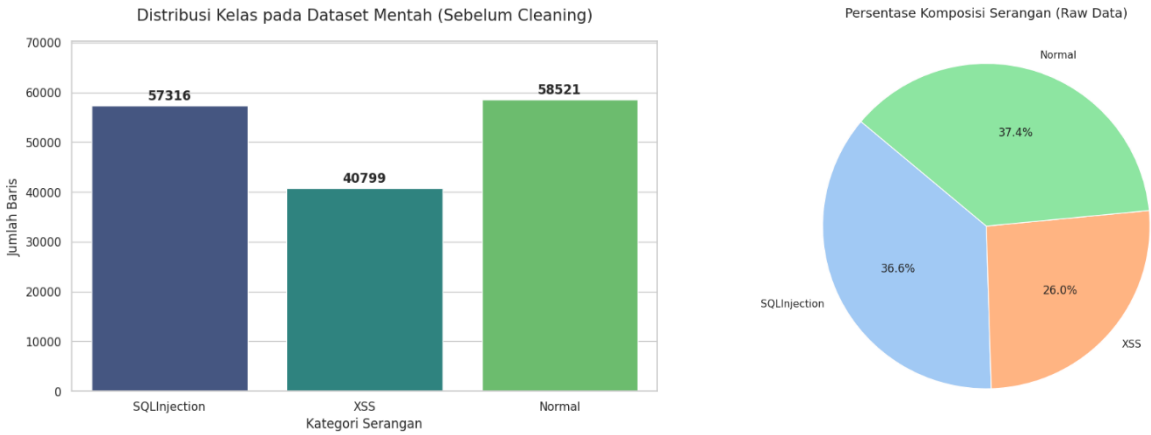

Figure 2. Raw Dataset Distribution Statistics Before the Cleaning Process

3.3 Data Pre-processing

The pre-processing stage focused on identifying and eliminating duplicate entries and null values that could introduce bias into the model. A total of 4,853 entries identified as duplicates or null values were removed from the raw dataset, producing a final dataset of 151,783 unique samples and achieving a data retention rate of 96.90%. The resulting distribution exhibited relative class balance — a factor of direct consequence for preventing algorithmic bias during training, given that skewed class proportions are known to systematically distort classifier decision boundaries.

3.4 Character-Level N-Gram Feature Extraction

Text representations of attack payloads were converted into numerical form using the character-level n-gram method. Unlike word-based extraction, this approach was selected for its ability to capture the finest structural fragments of an attack payload — including those deliberately obfuscated through encoding, spacing manipulation, or syntactic variation. The process involved decomposing each of the 151,783 entries into character-level fragments via TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, generating a high-dimensional feature space of 10,000 unique features. All feature mappings were stored in a binary file named char_vectorizer.pkl to ensure consistency during the testing phase.

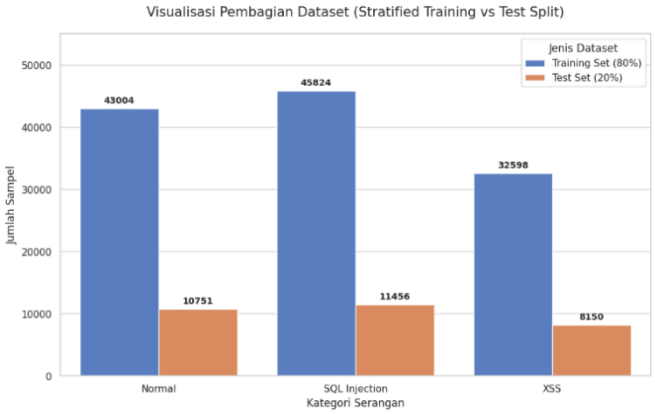

Figure 3. Character-Level N-Gram Feature Extraction Log

3.5 Ensemble Model Architecture

An ensemble learning architecture was employed by combining two classification algorithms: Random Forest and XGBoost. Random Forest reduces variance through bagging, while XGBoost minimizes bias through boosting — the two mechanisms are complementary by design, and their combination addresses weaknesses that either algorithm carries when deployed independently. The dataset was partitioned at an 80:20 ratio, yielding 121,426 training samples and 30,357 test samples. To ensure full experimental reproducibility, all algorithmic components were configured with explicit parameters. The TF-IDF Vectorizer operated at the character level (analyzer='char') with an n-gram range of 2 to 3 (ngram_range=(2, 3)) and a maximum feature limit of 10,000 dimensions (max_features=10000). The Random Forest classifier was constructed using 100 decision trees (n_estimators=100) with parallel computation enabled (n_jobs=-1). The XGBoost classifier was configured using a multi-class log loss evaluation metric (eval_metric='mlogloss'), with tree depth and learning rate set to framework defaults. Both models were combined through a Soft Voting strategy (voting='soft'), which aggregates class probability outputs from each base classifier to produce the final prediction. All model initialization and data partitioning steps used a fixed random seed of 42 (test_size=0.2, stratify by target label) to ensure metric consistency and reproducibility across runs. The complete configuration parameters are summarized in Table 2.

Table 2. Ensemble Model Configuration Parameters

System Component	Parameter	Configuration Value
TF-IDF Vectorizer	analyzer	'char'
	ngram_range	(2, 3)
	max_features	10,000
Random Forest	n_estimators	100
	n_jobs	-1
XGBoost	eval_metric	'mlogloss'
	max_depth & learning_rate	Default
Ensemble Voting	voting	'soft'
Data Partition & General	test_size	0.2
	stratify	Target Label
	random_state (Seed)	42

3.6 Evaluation and Validation Scheme

Model performance was measured using accuracy, precision, recall, and F1-score. ROC curve and AUC analysis was incorporated using the One-vs-Rest (OvR) approach for multi-class AUC calculation, with the Precision-Recall curve employed as a supplementary evaluation instrument — particularly relevant in security contexts where the cost of false negatives is asymmetric. A 5-Fold Cross-Validation scheme was applied to

ensure generalization capability and freedom from overfitting, with sequential execution adopted to maintain memory efficiency without compromising validation integrity.

3.7 Tools and Technologies

The experimental implementation was developed in Python. Core libraries included Scikit-learn for Random Forest and TF-IDF vectorization, and the XGBoost library for the gradient boosting classifier. The prototype system was built using the Flask web framework, and feature mappings were serialized using pickle (char_vectorizer.pkl) to ensure full reproducibility across experimental runs.

4. Result and Discussion

4.1 Results

4.1.1 Data Preparation and Pre-processing Outcome

The initial stage focused on ensuring data integrity through a rigorous curation process. The raw dataset, originally consisting of 156,636 samples, underwent cleaning procedures aimed at eliminating noise and data redundancy — a necessary step given that the data was aggregated from heterogeneous traffic log sources. A total of 4,853 entries identified as duplicate records and missing values were removed, producing a final dataset of 151,783 unique samples at a retention rate of 96.90%. The class distribution following cleaning consisted of 57,280 SQL Injection samples, 40,748 XSS samples, and 53,755 normal traffic samples — a relatively balanced proportion that directly reduces the risk of algorithmic bias during training. A detailed before-and-after comparison is presented in Table 3, and the resulting distribution is illustrated in Figure 4.

Table 3. Dataset Comparison Before and After the Cleaning Process

Data Category	Raw Data	Clean Data	Removed Data	Retention Rate
SQL Injection	57,316	57,280	36	99.94%
XSS	40,799	40,748	51	99.87%
Normal	58,521	53,755	4,766	91.86%
Total	156,636	151,783	4,853	96.90%

As shown in Table 3, the Normal class experienced the largest reduction, accounting for 4,766 of the 4,853 removed entries — indicating a higher concentration of duplicate and null records within that category. Figure 4 further illustrates the final class distribution following the pre-processing stage, confirming the relative balance achieved across all three categories.

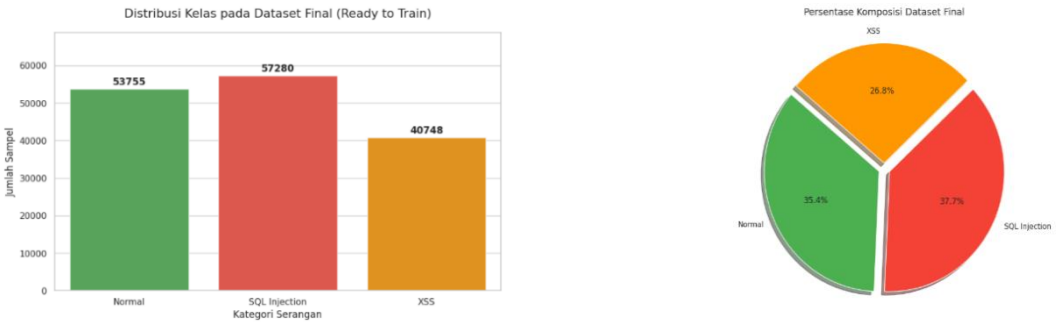

Figure 4. Distribution of the Final Dataset After Pre-processing

4.1.2 Character-Level Feature Extraction Analysis

Following data validation, the cleaned dataset was processed through character-level n-gram feature extraction. Each of the 151,783 entries was decomposed into the smallest character fragments via TF-IDF vectorization, generating a high-dimensional feature space of 10,000 unique features. Unlike word-level approaches, character-based decomposition enables the system to detect anomalies at the structural level of the payload — meaning the model remains effective even when attackers employ encoding, spacing manipulation, or syntactic obfuscation. All feature mappings were stored in a binary file named char_vectorizer.pkl to ensure consistency during the testing phase. Evidence of the successful extraction process is presented in Figure 5.

```
Starting Character-Level N-Gram feature extraction...
Extraction completed successfully!
Number of data rows: 151783
Number of unique n-gram features found: 10000
✅ Vectorizer saved to: /content/drive/MyDrive/research/sqli xss/char_vectorizer.pkl
```

Figure 5. Character-Level N-Gram Feature Extraction Log

4.1.3 Performance Evaluation Metrics

The ensemble model was evaluated against 30,357 test samples held out from the 121,426-sample training set. The model achieved an overall accuracy of 99.67%, with precision and recall values ranging from 0.99 to 1.00 across all three categories, as detailed in Table 4. The high precision and recall values confirm that the model accurately identifies attacks while rarely missing malicious activity — a critical requirement in security-sensitive deployment contexts where false negatives carry asymmetric consequences.

Table 4. Ensemble Model Classification Report

Data Category	Precision	Recall	F1-Score	Support
SQL Injection	1.00	0.99	1.00	11,456
XSS	1.00	1.00	1.00	8,150
Normal	0.99	1.00	1.00	10,751
Overall Accuracy			1.00	30,357

4.1.4 Visualization of Discriminative Power

To examine model performance at a granular level, a confusion matrix analysis was conducted to provide a transparent comparison between actual labels and model predictions. In the XSS category, 8,143 samples were correctly predicted, with only 6 misclassified as normal traffic and 1 as SQL Injection. For SQL Injection, 11,393 predictions were correct, with residual errors consisting of 61 samples misclassified as normal and 2 as XSS. Figure 6 presents the confusion matrix in absolute values, allowing the precise volume of correct and incorrect classifications to be identified across each category.

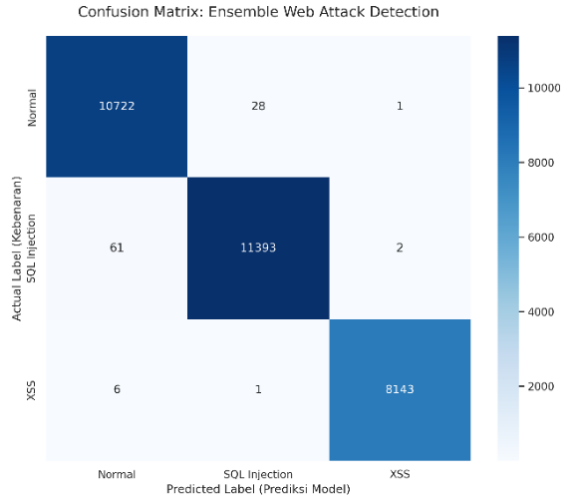

Figure 6. Web Attack Detection Confusion Matrix with Absolute Values

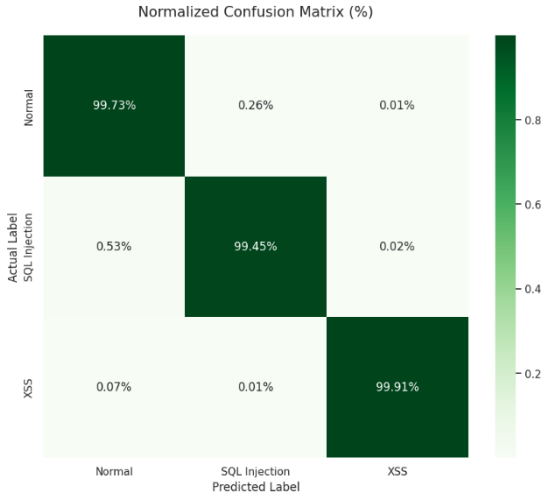

Figure 7. Confusion Matrix Normalized as Percentage

To evaluate these results relative to the total test data population, the confusion matrix is further expressed as normalized percentages in Figure 7, enabling a proportional comparison of classification performance across all three classes. The normalized results indicate that XSS achieved the highest per-class accuracy at 99.91%, followed by normal traffic at 99.73%, where only 0.26% (28 samples) were incorrectly classified as SQL Injection — errors attributable to structural similarities between certain normal HTTP requests and SQL syntax patterns. To further validate the model's discriminative capability across varying classification thresholds, ROC curve analysis was conducted using the One-vs-Rest (OvR) approach. Figure 8 presents the resulting multi-class ROC curves alongside their corresponding AUC values.

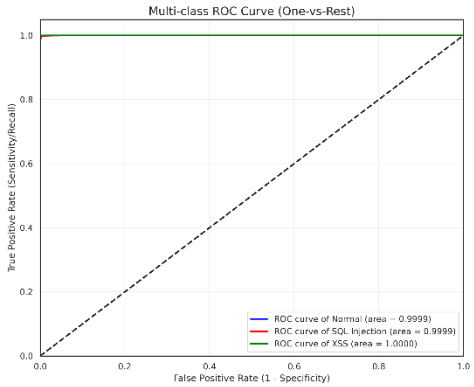

Figure 8. Multi-class ROC Curve

The ROC analysis yielded AUC values of 1.0000 for XSS and 0.9999 for both SQL Injection and normal categories — indicating near-perfect separability between classes. As a complementary evaluation, Precision-Recall curves were additionally analyzed to assess the stability of precision across varying recall levels, particularly in the context of class imbalance sensitivity. Figure 9 presents the resulting curves for all three categories.

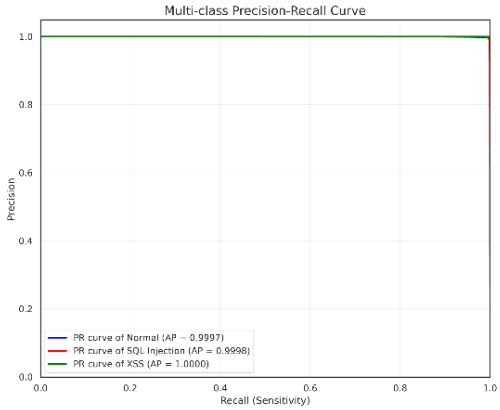

Figure 9. Multi-class Precision-Recall Curve

The Precision-Recall analysis shows Average Precision (AP) values of 1.0000 for XSS, 0.9998 for SQL Injection, and 0.9997 for the normal category. The curves consistently remain near the upper-right corner, confirming that the model maintains very high precision even at maximum recall — a behavior that directly reduces the risk of false alarms disrupting web application operations. Taken together, the low off-diagonal misclassification rates and near-perfect AUC and AP scores collectively confirm that character-level n-gram features are highly effective in distinguishing the structural characteristics of each attack type.

4.1.5 Statistical Stability and Validation

To verify that the model's performance is not an artifact of a favorable data partition, a 5-Fold Cross-Validation scheme was applied sequentially to maintain memory efficiency. The dataset was divided into five distinct subsets, with each subset alternately serving as the test set while the remaining four were used for training. Results across all five folds demonstrated strong consistency, with accuracy ranging narrowly from 0.9959 to 0.9968, as detailed in Table 5.

Table 5. 5-Fold Cross-Validation Results

Fold	Accuracy Score
Fold 1	0.9966
Fold 2	0.9959
Fold 3	0.9966
Fold 4	0.9968
Fold 5	0.9961
Mean CV Accuracy	0.9964
Standard Deviation	0.0003

The mean cross-validation accuracy of 99.64% sits within 0.03 percentage points of the test accuracy of 99.67% — a negligible gap that indicates the model generalizes well to unseen data rather than fitting to the specific characteristics of a single partition. More telling still is the standard deviation of 0.0003, a value approaching zero that confirms minimal variance in model performance across data splits. Evidence of the sequential cross-validation process is presented in Figure 10.

```
Starting 5-Fold Cross Validation sequentially
Processing Fold 1...
Fold 1 Result: Accuracy = 0.9966
Processing Fold 2...
Fold 2 Result: Accuracy = 0.9959
Processing Fold 3...
Fold 3 Result: Accuracy = 0.9966
Processing Fold 4...
Fold 4 Result: Accuracy = 0.9968
Processing Fold 5...
Fold 5 Result: Accuracy = 0.9961

==============================
Average CV Accuracy: 0.9964
Standard Deviation: 0.0003
==============================
```

Figure 10. Sequential Log of 5-Fold Cross-Validation Results

4.1.6 Prototype Implementation and Real-Time Validation

To evaluate operational capability beyond offline metrics, a web anomaly detection prototype was implemented using the Flask framework. The system classifies payloads in real time and provides inference transparency through an Explainable Artificial Intelligence (XAI) interface, as illustrated in Figure 11. In addition to prediction labels (Normal, SQL Injection, or XSS), the system presents a Confidence Score and a character vector fragmentation visualization (N-Gram Chunks) — technically demonstrating how a payload is decomposed into high-dimensional feature vectors before reaching the ensemble classifier. When a payload is classified as an attack, the system renders a knowledge-base panel containing threat anatomy and mitigation recommendations, including Parameterized Queries and secure input sanitization practices. This educational layer reflects a deliberate design choice: the framework is not purely a classification instrument, but a practical cybersecurity tool oriented toward actionable problem-solving.

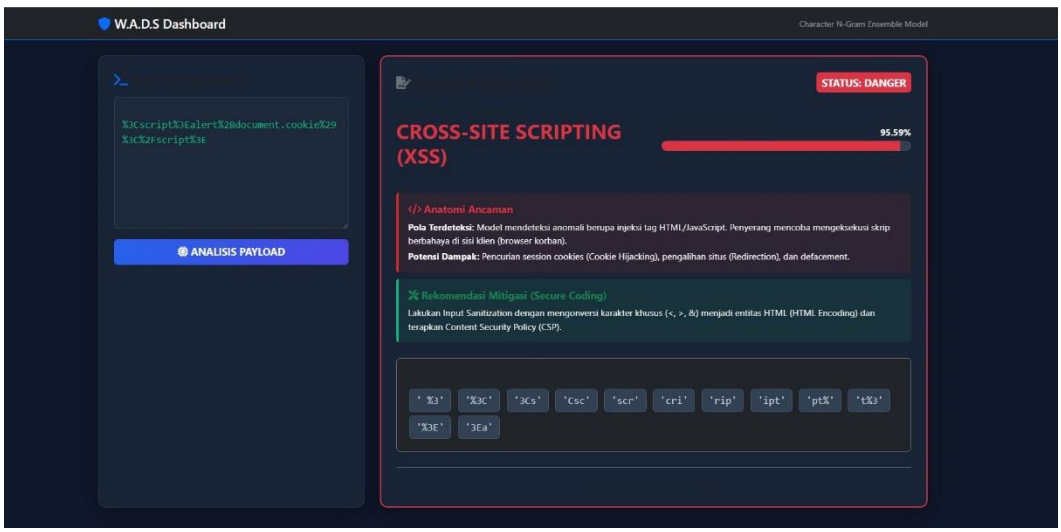

Figure 11. Interface of the Ensemble Learning-Based Web Attack Detection System

Black-box validation was subsequently conducted using external payload variations never seen during training, with particular emphasis on obfuscated inputs. All results were automatically recorded in the application's internal database, as presented in Table 6. Confidence scores exceeded 90% across nearly all attack payloads, including structurally complex cases such as time-based blind SQL Injection and URL-encoded XSS variants. Normal inputs — including Indonesian-language academic text and proper names — were consistently

classified correctly, with none triggering false positives. The model's ability to handle obfuscated characters without misclassifying benign inputs provides direct evidence that the algorithm generalizes anomaly patterns at the character distribution level, rather than memorizing attack signatures in the manner of traditional rule-based systems.

Table 6. Experimental History and Black-Box Validation Using Unseen Data

Payload Data	Prediction Result	Confidence (%)
' OR 1=1 LIMIT 1 OFFSET 0 --	SQL Injection	84.42
' UNION SELECT null, username, password FROM admin_users--	SQL Injection	89.45
' OR 'a' = CONCAT('a','')	SQL Injection	89.49
1' AND (SELECT * FROM (SELECT(SLEEP(5)))a) AND '1'='1	SQL Injection	89.49
' OR LOWER('A')='a' --	SQL Injection	91.38
' oR 'a'='a	SQL Injection	98.96
' OR '1'='1	SQL Injection	99.16
' OR 5 > 2 --	SQL Injection	99.81
' OR 1.2e0 = 1.2 --	SQL Injection	100
%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E	Cross-Site Scripting (XSS)	95.59
<details/open/ontoggle=alert(1)>	Cross-Site Scripting (XSS)	99
<svg><script xlink:href="data:,alert(1)" />	Cross-Site Scripting (XSS)	99.5
<img src=x onerror=alert(String.fromCharCode(88,83,83))>	Cross-Site Scripting (XSS)	99.5
<svg onload=alert(document.cookie)>	Cross-Site Scripting (XSS)	99.5
<img src="x" onerror="alert('XSS')">	Cross-Site Scripting (XSS)	99.5
<script>alert(XSS)</script>	Cross-Site Scripting (XSS)	99.98
<scr%00ipt>alert(1)</script>	Cross-Site Scripting (XSS)	99.99
<script>alert(1)</script>	Cross-Site Scripting (XSS)	99.99
Pengumuman: Jadwal ujian akhir semester telah dirilis.	Normal	85
cara mendaftar beasiswa mahasiswa berprestasi 2026	Normal	85.48
Mohammad Bayu Anggara	Normal	89.26
Mohammad Bayu Anggara, S.Kom., M.Kom.	Normal	93.26
"Cara belajar machine learning untuk pemula"	Normal	93.99
Final exam schedule for the odd semester Faculty of Information Technology	Normal	96.44

The results presented in Table 6 collectively demonstrate that the model maintains high classification confidence across all three categories — including structurally obfuscated and encoding-manipulated attack payloads — without generating false positives on benign inputs. This consistency across diverse payload types confirms that the proposed ensemble framework possesses strong generalization capability and is operationally viable as a real-time web attack detection layer in production-scale web application environments.

4.2 Discussion

The results confirm that character-level n-gram features combined with ensemble learning produce a highly accurate and stable web attack detection framework. The near-perfect ROC AUC scores (1.0000 for XSS, 0.9999 for SQLi) and consistently high cross-validation accuracy (99.64% mean) collectively indicate that the model captures genuine structural discriminants of attack payloads rather than overfitting to specific training samples. The duplicate-removal pre-processing step was critical in this regard — eliminating 4,853 redundant entries that would otherwise have introduced memorization bias into the classifier. Compared to prior work, the proposed framework advances beyond single-classifier approaches such as those evaluated by Le *et al.* (2024), wherein XGBoost alone showed limitations in SQL Injection detection. The combination of Random Forest's variance-reduction via bagging and XGBoost's bias-reduction via boosting produces

complementary strengths that neither algorithm achieves independently — consistent with the theoretical and empirical arguments advanced by Lower & Zhan (2020). Bakır (2025), employing the UniEmbed multi-level feature fusion approach combining Word2Vec, Universal Sentence Encoder, and FastText, reported a best accuracy of 99.82% for XSS detection using the MLP classifier and accuracy rates exceeding 99.80% for SQL Injection detection across two datasets. Tadhani *et al.* (2024), using a hybrid CNN-LSTM architecture, achieved accuracies of 99.84%, 99.23%, and 99.77% on the SQLi-XSS Payload, Testbed, and HTTP CSIC 2010 datasets respectively. Both studies report competitive or marginally higher accuracy figures — yet they rely on significantly more complex architectures that carry substantially higher computational costs. The proposed ensemble framework achieves comparable accuracy through a computationally lighter approach, where the performance gain stems directly from the syntactic information retained by character n-gram decomposition of obfuscated payloads — precisely the aspect that word-level methods consistently fail to address. The URL-encoded payload (%3Cscript%3E) classified with 95.59% confidence in black-box testing directly illustrates this obfuscation resilience. The primary limitation, however, is the high dimensionality of the feature space (10,000 features), which introduces memory and computational overhead for sparse matrix processing in high-throughput real-time deployment scenarios. The use of a single publicly available dataset from Kaggle, while enabling clean variable isolation for validating the 10,000-dimensional feature extraction pipeline, also constrains generalizability claims to traffic patterns not represented in that source — a limitation that future work should address through multi-source dataset expansion.

5. Conclusion and Recommendations

This study demonstrates that the integration of character-level n-gram feature extraction with an ensemble learning framework combining Random Forest and XGBoost constitutes a highly effective solution for simultaneous SQL Injection and XSS attack detection. The proposed model achieved an overall test accuracy of 99.67%, with precision and recall values reaching 1.00 for both XSS and SQL Injection categories. Statistical generalizability was confirmed through 5-fold cross-validation, yielding a mean accuracy of 99.64% and a standard deviation of 0.0003 — establishing the framework as robust and free from overfitting. ROC AUC scores of 1.0000 for XSS and 0.9999 for SQL Injection further demonstrate near-perfect discriminative capability across all classification thresholds. Black-box validation on unseen obfuscated payloads, including URL-encoded and structurally manipulated attack strings, confirmed that the model generalizes anomaly patterns at the character distribution level rather than memorizing fixed attack signatures — a distinction that is critical for deployment in dynamic, adversarial environments.

The primary contribution of this research is the development of a combined Random Forest and XGBoost ensemble framework with character-level n-gram feature extraction, specifically validated for simultaneous multi-class web attack detection across SQL Injection, XSS, and normal traffic categories. Unlike prior approaches that address SQLi or XSS in isolation, the proposed framework achieves competitive accuracy with greater computational accessibility — without incurring the architectural complexity of deep learning alternatives such as CNN-BiLSTM or multi-level NLP embedding fusion. The use of 10,000 character-level features enables granular structural analysis of attack payloads, providing demonstrated resilience against obfuscation techniques validated empirically through black-box testing on unseen encoded inputs. The framework further contributes a Flask-based prototype with Explainable AI (XAI) capabilities, offering transparency into the inference process alongside actionable mitigation recommendations — positioning it not merely as a classification instrument, but as a practical cybersecurity tool oriented toward operational deployment.

The primary limitation of this study is the high-dimensional feature space (10,000 features), which may introduce memory and computational overhead in high-throughput real-time deployment scenarios where sparse matrix processing at scale becomes a bottleneck. Additionally, the dataset originates from a single publicly available source, which may limit generalizability to novel attack variants or domain-specific traffic patterns not represented in the training data. The black-box validation, while demonstrating strong obfuscation resilience, was conducted on a limited set of manually crafted payloads — broader validation against real-world production traffic logs would further strengthen confidence in deployment readiness.

Future research should explore dimensionality reduction techniques such as Truncated Singular Value Decomposition (SVD) or feature importance-based selection to reduce computational overhead while preserving detection accuracy. Expanding the training dataset with diverse, real-world traffic logs from multiple sources would directly address the generalizability constraint identified above and expose the model to a broader range of obfuscation strategies and attack variants. Investigating the framework's performance against advanced polymorphic and multi-stage attack chains represents an important next step in validating production-grade robustness. The ensemble architecture could further be extended to include additional gradient boosting classifiers such as LightGBM or CatBoost, or integrated with attention-based mechanisms to

improve classification of structurally complex payloads at the boundary between attack categories. Finally, deployment evaluation under production-scale traffic conditions — including latency and throughput benchmarking — is essential prior to integration into live web application security infrastructure, ensuring that the framework's offline accuracy translates reliably into real-time operational performance.

References

Ahmed, M., & Uddin, M. N. (2020). Cyber attack detection method based on NLP and ensemble learning approach. *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, 1–6. https://doi.org/10.1109/ICCIT51783.2020.9392682

Bakır, R. (2025). UniEmbed: A novel approach to detect XSS and SQL injection attacks leveraging multiple feature fusion with machine learning techniques. *Arabian Journal for Science and Engineering*, *50*(19), 15591–15604. https://doi.org/10.1007/s13369-024-09916-4

Feng, Y., Yang, Z., Sun, Q., & Liu, Y. (2024). SEDAT: A stacked ensemble learning-based detection model for multiscale network attacks. *Electronics*, *13*(15), Article 2953. https://doi.org/10.3390/electronics13152953

Habibi, G., & Surantha, N. (2020). XSS attack detection with machine learning and n-gram methods. *2020 International Conference on Information Management and Technology (ICIMTech)*, 516–520. https://doi.org/10.1109/ICIMTech50083.2020.9210946

Işiker, B., & Soğukpınar, İ. (2021). Machine learning based web application firewall. *2021 2nd International Informatics and Software Engineering Conference (IISEC)*, 1–6. https://doi.org/10.1109/IISEC54230.2021.9672335

Kiruthika, S., Roshni, A., & Padmavathi, G. (2024). Detection of Obfuscated Malware using Ensemble Learning Techniques. *Grenze International Journal of Engineering & Technology (GIJET)*, *10*.

Kshirsagar, D., & Kumar, S. (2020). An ensemble feature reduction method for web attack detection. *Journal of Discrete Mathematical Sciences and Cryptography*, *23*(2), 515–529. https://doi.org/10.1080/09720529.2020.1721861

Le, T., Hwang, Y., Choi, C., & Wardhani, R. W. (2024). Enhancing SQL injection detection with trustworthy ensemble learning and boosting models using local explanation techniques. *Preprints*, Article 2024100878. https://doi.org/10.20944/preprints202410.1878.v1

Li, Z., Liu, F., Gu, Z., & Liu, Y. (2025). XSS attack detection method based on CNN-BiLSTM-Attention. *Applied Sciences*, *15*(16), Article 8924. https://doi.org/10.3390/app15168924

Lower, N., & Zhan, F. (2020). A study of ensemble methods for cyber security. *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, 1001–1009. https://doi.org/10.1109/CCWC47524.2020.9031256

Odeh, A., & Taleb, A. A. (2024). Ensemble learning techniques against structured query language injection attacks. *Indonesian Journal of Electrical Engineering and Computer Science*, *35*(2), 1004–1012. https://doi.org/10.11591/ijeecs.v35.i2.pp1004-1012

Panadiya, P., & Singhal, M. K. (2024). Advanced detection and prevention of SQL injection attacks using machine learning techniques for enhanced web security. *International Journal of Scientific Research in Science and Technology*, *11*(6), 554–564.

Patil, S., & Bansode, R. (2024). Advancing web security: Machine learning-based attack detection with optimized features. *Panamerican Mathematical Journal*, *35*(2s), 571–579. https://doi.org/10.52783/pmj.v35.i2s.2938

Phuong, T. T., The, T. T., Shigetomi, R., Yamamura, Y., & Nakata, T. (2020). Boosting homograph attack classification using ensemble learning and N-gram model. *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 1983–1988. https://doi.org/10.1109/TrustCom50675.2020.00271

Ren, X., Hu, Y., Kuang, W., & Souleymanou, M. B. (2018). A web attack detection technology based on bag of words and hidden Markov model. *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 526–531. https://doi.org/10.1109/MASS.2018.00081

Riadi, I., Umar, R., & Lestari, T. (2020). Analisis kerentanan serangan cross site scripting (XSS) pada aplikasi smart payment menggunakan framework OWASP. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, *5*(3), 146–152.

Sonoda, M., Matsuda, T., Koizumi, D., & Hirasawa, S. (2011). On automatic detection of SQL injection attacks by the feature extraction of the single character. *Proceedings of the 4th International Conference on Security of Information and Networks (SIN '11)*, 81–86. https://doi.org/10.1145/2070425.2070440

Sornsuwit, P., & Jaiyen, S. (2019). A new hybrid machine learning for cybersecurity threat detection based on adaptive boosting. *Applied Artificial Intelligence*, *33*(4), 1–21. https://doi.org/10.1080/08839514.2019.1582861

Subba, B., & Gupta, P. (2021). A TF-IDF vectorizer and singular value decomposition based host intrusion detection system framework for detecting anomalous system processes. *Computers & Security*, *100*, Article 102084. https://doi.org/10.1016/j.cose.2020.102084

Tadhani, J. R., Vekariya, V., Sorathiya, V., Alshathri, S., & El Shafai, W. (2024). Securing web applications against XSS and SQLi attacks using a novel deep learning approach. *Scientific Reports*, *14*, Article 1228. https://doi.org/10.1038/s41598-023-48845-4.