International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 13– No.4, August 2026 – www.ijais.org

A Stacking Ensemble Approach for the Detection of Cross-Site Scripting and Cross-Site Request Forgery Attacks

E.O. Oginni
Department of Software Engineering
Obafemi Awolowo University
Ile-Ife, Osun State

A.O. Amoo
Department of Software Engineering
Obafemi Awolowo University
Ile-Ife, Osun State

T.O. Omodunbi
Department of Information Systems
Obafemi Awolowo University
Ile-Ife, Osun State

A.S. Afolayan
Department of Software Engineering
Obafemi Awolowo University
Ile-Ife, Osun State

ABSTRACT
The growing use of web-based applications has simultaneously increased vulnerability to security attacks, in particular Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF), which are among the most common online vulnerabilities. In this research, a stacking ensemble approach was used to detect XSS and CSRF attacks using structured web request data, with labelled datasets of normal and malicious web traffic obtained via publicly available GitHub repositories. It integrates three different base learners, such as the Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP) and uses an XGBoost meta-learner to boost the probabilistic outputs of the base learners to improve the detection results. The proposed model is a combination of heterogeneous classifiers, which improves the detection accuracy and robustness. The experimental results reveal that the stacking ensemble model is an effective solution for detecting web attacks and is more effective than the base learners. The model had an accuracy of 95%, and had a precision of 95%, showing increased detection reliability and a reduced false positive rate. Moreover, it has a high discriminative ability as it has an ROC-AUC score of 0.9808. The results highlight the importance of the stacking ensemble approach in enhancing the robustness and stability of machine learning-based software systems for protecting web applications from XSS and CSRF attacks. This study focuses on XSS and CSRF vulnerabilities and the application of stacking ensemble learning in the detection of attacks on web applications.

General Terms
Web Application Security, Intrusion Detection, Cybersecurity and Machine Learning.

Keywords
Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), Stacking Ensemble Learning, Web Application Security, Machine Learning-Based Attack Detection.

1. INTRODUCTION
Web technologies have undergone a tremendous evolution, changing the way that people, businesses, and even organizations communicate, share information, and transact online [1]. The modern web applications provide many different services, such as online banking, e-commerce, cloud computing, and social networks [2]. As the reliance on these applications continues to grow, so does the need to protect them against increasingly sophisticated cyber threats that may compromise the confidentiality, integrity, and availability of sensitive information [3].

Among the most critical security threats to web applications are Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF), both of which remain prominent vulnerabilities in the Open Worldwide Application Security Project (OWASP) 2025 report [4] [5]. These attacks target application code and input-validation functions to inject malicious code, steal user sessions, or do unauthorised access or mandate on behalf of authorised users. XSS attacks happen when malicious scripts are inserted into web pages and then executed in a browser of a victim [6], while CSRF attacks are used to force authenticated users to unwittingly take unintended actions using a trusted web application [7].

Traditionally, the identification and prevention of web vulnerabilities have been done by conventional security measures, which include firewalls, rule-based detection systems and manual code reviews [8]. These techniques, however, are heavily signature- and rule-dependent and are not very useful for the creation and evolution of today's advanced attack patterns. In recent years, machine learning techniques have gained increasing attention in cybersecurity due to their ability to analyse large volumes of data, learn patterns from network traffic, and identify anomalous or malicious behaviour [9], [10]. Several machine learning algorithms, including Support Vector Machines (SVM), Random Forest (RF), and Artificial Neural Networks (ANN), have been applied to intrusion detection and cyberattack classification [9],[11]. Among neural network-based approaches, Multilayer Perceptrons (MLP) have also been employed for the detection and classification of malicious network activities [11], [12].

While individual machine learning models can be effective, they can have varying results because they learn patterns in data in different ways. Consequently, not all models will necessarily be

the most accurate. This drawback can be addressed by ensemble learning methods, which fuse multiple classifiers to make better predictions. Among the available ensemble techniques, stacking employs several base learners whose outputs are integrated through a meta-learner to generate more accurate predictions than those obtained from individual models.

This study proposes a stacking ensemble framework to detect Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) attacks in web applications**.** The proposed framework combines three base learners, while Extreme Gradient Boosting (XGBoost) serves as the meta-learner. By integrating the strengths of these classifiers, the framework aims to provide a more robust and reliable approach for detecting multiple web application attacks within a single predictive model.

Therefore, this study contributes to knowledge through the following:

i. Introducing a unified stacking ensemble model for the detection of Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) attacks within a single classification framework, enabling earlier detection of attacks, enhancing the security and reliability of web applications.

ii. Complementing the work of many existing studies that address XSS and CSRF attacks separately with a single machine-learning-based model to detect both attacks, which provides a more holistic solution for web application security.

iii. Demonstrating the effectiveness of a stacking ensemble architecture, thereby improving detection performance and enhancing the robustness of automated web attack detection.

2. LITERATURE REVIEW

A brief overview of Web application attacks which were conceived for this study is provided in this section: Cross-Site Scripting and Cross-Site Request Forgery. They are among the most common vulnerabilities in today's Web applications and can be quite harmful to users' privacy and system security.

2.1 Cross-Site Scripting

Cross-Site Scripting (XSS) is a type of injection vulnerability in web applications in which an attacker injects malicious scripts into otherwise trusted web content. When the vulnerable application delivers the injected content to a user, the victim's browser may execute the malicious script as though it originated from the trusted application [13]. This is a common vulnerability which arises when web application does not validate or sanitize user inputs before displaying them on web pages [14]. This means that attackers can run malicious scripts that could steal sensitive data, including session cookies, authentication tokens or other user data [15].

XSS continues to be One of the most common and severe web application vulnerabilities that can affect user sessions and lead to users being redirected to malicious sites, and steal sensitive information [16].

Figure 1 presents how an XSS attack can occur in a web application. First, the attacker enters a malicious JavaScript code through an input field that accepts user data, such as a search box, comment section, or registration form. If the application does not properly check the input or encode the output, the malicious code may be included in the web page sent back to the user. When the page is opened by the targeted user, the browser runs the injected script as part of the trusted application. Depending on the attack, the attacker may use this to steal session information, obtain sensitive data, change the content of the page, or redirect the user to another website. Proper input validation, output encoding, and secure coding practices are crucial in minimizing the risk of XSS attacks in web applications.

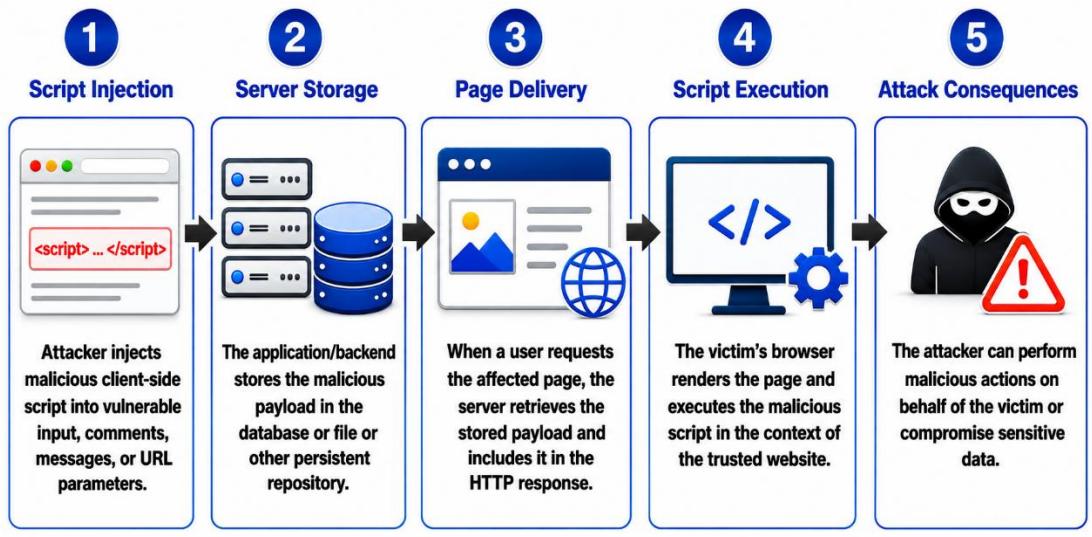

Fig 1: XSS Attack Data Flow

2.2 Cross-Site Request Forgery (CSRF)

CSRF is a type of attack in which an attacker tricks a user into making an unwanted request to a web application, which can lead to unwanted actions being taken by the user [17]. In a CSRF attack, the malicious party requests a user who has already signed in to make an unexpected request to the application [18].

The attacks are usually made via malicious links, scripts or embedded requests in web pages or emails. When the user interacts with such content, the browser automatically sends authentication information (session cookies) with the request, which may enable the attacker to perform unauthorized operations with the user's active session. These actions may include modifying account settings, submitting forms, or performing financial transactions [19].

Figure 2 shows the general steps of a CSRF attack. The intrusion starts when a legitimate user logs in to a trusted web application**,** and then the server creates an authenticated session for the user by sending a session cookie to the user's browser.

The attacker then submits a request to the web application for a sensitive operation, like changing account settings or transferring money, after which the attacker lures the authenticated user into clicking a link or visiting a malicious Web site that includes the forged request. If the user is already logged in, the web application automatically adds the valid session cookie to the requests made to it by the browser. The server cannot distinguish between the forged request and the real request, and it acts as if it had come from the person logged on and carries out the required action. This means that unauthorized operations are executed without the user's knowledge or consent, which can result in data modification, unauthorized transactions, account compromise, or other security issues. It is important to have proper request validation mechanisms, such as anti-CSRF tokens, SameSite cookies, and origin verification, because CSRF attacks rely on legitimate user authentication rather than direct system compromise; due to this, they can bypass many traditional security mechanisms. Effective mitigation, therefore, requires additional protections such as anti-CSRF tokens, request validation techniques, and behavioural monitoring mechanisms.

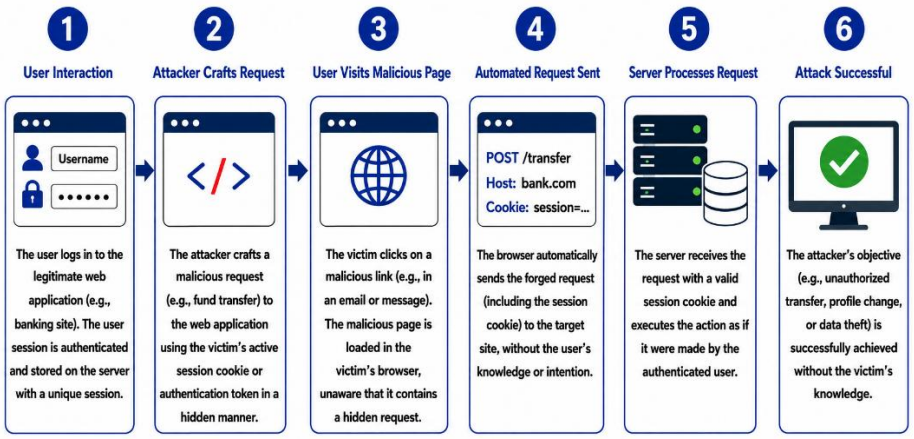

Fig 2: CSRF Attack Data Flow

2.3 Review of Machine-Learning Related Works

As web application attacks become more sophisticated, traditional rule-based security mechanisms may not always be effective in detecting newer and more complex threats [20]. With the increasing number of web security threats, researchers have been looking into machine learning and deep learning for automatic detection. XSS and CSRF are still prevalent in web apps, posing persistent security risks. As a result, different machine learning algorithms have been investigated to improve the identification of these threats.

Early machine learning approaches focused primarily on XSS detection. MLPXSS was proposed by Mokbal *et al.* [21] that uses dynamic feature extraction to detect obfuscated and non-obfuscated XSS payloads by employing a Multilayer Perceptron (MLP) model. Analysed HTML, JavaScript, and URL features of web requests in the model. The model achieved 99.32% accuracy and AUC-ROC of 99.02 on a dataset of 138,569 samples, outperforming other classifiers.

Likewise, Stency *et al.* [22] tested multiple supervised classification approach with a labelled collection of benign and malicious URLs retrieved from Kaggle. The best model that performed well among the evaluated models was the Random Forest classifier with 97.61% of accuracy, which proved that ensemble learning techniques are useful in detecting web-based attacks.

To further enhance the detection performance, Mokbal *et al.* [23] presented a detection framework named XGBXSS that combines two enhancements: an Enhanced Feature Extraction (EFE) method and a hybrid feature selection method based on Information Gain and Sequential Backward Selection (SWS) technique, with the base detection model being XGBoost. The proposed method resulted in 30 significant features and 99.59% accuracy, 0.20% false positive, 0.98% false negative and AUC of 99.41.

The detection of XSS using deep learning techniques has also been examined. Kumar *et al.* [24] presented a Convolutional Neural Network (CNN) model which was able to directly process the ASCII representation of scripts without any complicated preprocessing steps to attain a model accuracy of 98%. Likewise, Alaoui [25] proposed DeepWAF, a model that uses the LSTM network and Word2Vec embeddings to analyse the HTTP request and reached an accuracy of 95.2%.

Hybrid architectures and ensemble have also been studied recently. Tadhani et al. [26] used CNN–LSTM hybrid model for SQL Injection and XSS attack detection achieving 99.77% accuracy. Likewise, Muralitharan *et al.* [27] designed a stacking ensemble deep neural network, combining traditional machine learning models with deep learning architectures to get a 99.82% F1 score and 99.5% accuracy for detecting XSS. In another study, Alhamyani and Alshammari [28] evaluated multiple classifiers, including RF, SVM, MLP and CNN, on a dataset containing 138,569 XSS samples with 167 features. Based on the results from the feature selection processes (Information Gain and ANOVA), the RF classifier was found to be the top-performing model with an accuracy of 99.78% and a ROC-AUC score of 100%.

Studies of automated detection of CSRF vulnerabilities have been comparatively limited, compared to XSS detection studies. Nagpal *et al.* [29] have developed a multi-phase security engine named SECSIX that is capable of detecting SQL Injection, XSS, and CSRF attacks in PHP applications, through static analysis, during runtime, and by using attack signatures, respectively. The framework, however, was targeted towards PHP environments only and did not offer any quantitative evaluation metrics.

To overcome this problem, Pellegrino *et al.* [30] proposed an automated dynamic analysis framework called Deemon, which detects CSRF vulnerabilities by tracing runtime information of HTTP interactions, server-side logic and database queries. Then, Calzavara et al. [31] presented Mitch, the first machine-learning based framework for the detection of CSRF**.** In this work, Mitch uses a Random Forest (RF) classifier to categorize HTTP requests into sensitive and non-sensitive to get ROC-AUC of 0.932. The method is mostly based on the user's interactions that can be detected by the browser, however, and may not cover vulnerabilities on the server.

Further improvements were introduced by Hadavi and Sadeghi [32], who proposed a passive and language-agnostic detection approach that analyses browser–server communication to identify missing protection mechanisms such as anti-CSRF tokens and SameSite cookies. When tested on 22 real-world web applications that included banking and e-commerce platforms, Ismail and Hassan [33] were able to achieve a detection accuracy of around 91% with their automated detection system using web scraping and token analysis. In recent years, Sravani *et al.* [34] proposed a black-box machine learning based CSRF attack detection method by examining the pattern of the traffic in the HTTP protocol with 85% accuracy, 0.78 precision, and 0.85 recall.

There have been some attempts to fix both XSS and CSRF. Kshetri *et al.* [35] developed an algorithmic XSSF framework based on the six-stage cybersecurity lifecycle of recognition, examination, evaluation, protection, monitoring and recovery for detection and analysis of XSS and CSRF attacks. The framework incorporates several machine learning models, but specific performance parameters were not included.

In summary, the existing studies revealed that machine learning and deep learning techniques can be effective in the detection of Web attacks. Most methods, however, only deal with one attack type, and few models deal with both XSS and CSRF attacks simultaneously. This restriction underscores the need for stronger and more scalable detection systems that can detect multiple web-based threats in a single system.

3. METHODOLOGY

In this study, a machine learning-based detection model for XSS and CSRF attacks in web applications was developed. The dataset used was a collection of benign and malicious samples labeled in web request logs with different attack patterns. The collected data was subjected to several data processing techniques, such as data cleaning, feature selection, and data normalization to improve the quality and consistency of the dataset.

The processed data was then divided into a training set and a testing set for training and testing the model. A stacking ensemble learning method was used to construct the detection model. The first layer consisted of three base classifiers which were trained to learn different patterns in the web request data. The predictions generated by these base learners then serves as the input for a meta-learner. At the meta-learning stage, the Extreme Gradient Boosting algorithm (XGBoost) was employed as the meta-learner to combine the predictions from the base classifiers and generate the final classification output.

3.1 Datasets Description

A publicly available labelled Web traffic dataset was used for the development of the classifier model for detecting CSRF and XSS attacks. The CSRF dataset was compiled from a GitHub repository (https://github.com/alviser/mitch), and consists of a collection of 6,312 samples of HTTP requests corresponding to Web traffic behaviour. The class for attack refers to malicious CSRF activities that have been grouped together in the dataset; the class for benign refers to normal web traffic. These records include ordered information like HTTP methods used, number of parameters passed, indicators of authentication tokens in the header, and patterns that have been observed for sensitive keywords in the request.

These features are employed to describe the behavioral patterns linked with CSRF vulnerabilities and to facilitate the machine learning model's effective learning.XSS dataset is obtained from a public git hub repository (https://github.com/fawaz2015/XSS-dataset) which has 138,567 samples of HTTP requests. It consists of benign and malicious web requests represented using 66 extracted features. These features detect suspicious HTML attributes, event handlers, JavaScript functions, and HTML tags that are likely indicators of an XSS attack. Both datasets are organised in a machine-learning-ready format and can be directly used in classification models. They also serve as a strong foundation for training and testing the proposed stacking ensemble model that combines MLP, SVM and Random Forest with XGBoost as a meta-learner for enhanced detection of web-based attacks.

3.2 Data Preprocessin

The raw data was subjected to multiple pre-processing steps to guarantee data quality, consistency, and appropriateness for classification using machine learning. These CSRF and XSS datasets were first merged to form a unified dataset for the development of a combined web attack detection model. All the HTTP request samples were cleaned from the data source by removing duplicate data, missing values, and corrupted data so that only valid HTTP requests remain for analysis. The next step was to translate the original labels of both datasets into a common binary label system: benign requests were labelled as 0, and malicious requests were labeled as 1. The mapping was automatically completed to be consistent throughout the merged dataset. No further encoding was needed since all the features extracted were already numeric. The data was,

normalized using StandardScaler for feature scaling to ensure that features have a uniform distribution, thus enhancing the performance and convergence of the learning algorithms. To ensure a balanced representation of CSRF and XSS data sets, undersampling was used for harmonization. This was done to ensure that both datasets contributed equally to the model, with no one dataset dominating over the other while training the model, after which the processed data was then divided into a training set and a test set, using an 80/20 split. Moreover, to improve the generalization capability of the model and develop a strong evaluation system, the k-fold cross-validation technique was employed during the training process. All pre-processing was performed in Python, with Pandas for data manipulation and Scikit-learn for data scaling and model preparation, while dataset harmonization operations were performed using Imbalanced-learn.

3.3 Proposed Stacking Ensemble Framework

To enhance the detection of XSS and CSRF attacks, a stacking ensemble learning framework was developed. It is based on three heterogeneous base learners and a meta-learner. The proposed framework takes advantage of multiple classifiers to achieve better detection accuracy, robustness and generalization performance. The architecture of the proposed framework is shown in Figure 3. The framework starts with the data preprocessing stage, where the dataset was cleaned harmonized and normalized. The normalized data is then passed into the first layer of the stacking model, which consists of three separate base learners. To avoid overfitting and improve robustness, the Random Forest model was trained with an ensemble of decision trees built from various subsets of the training data set. The Support Vector Machine was implemented with Radial Basis Function (RBF) kernel to learn the complex non-linear relationships between the web request features and probability estimation was turned on to generate the required probabilistic outputs for stacking. The Multi-Layer Perceptron was constructed as a feedforward network with the Adam optimization method and backpropagation training algorithm for learning complex attack patterns. This study used 5-fold cross-validation to produce out-of-fold predictions for each base learner to build the stacking ensemble. For each fold, the base models were trained on four partitions and validated on the remaining partition. The probabilistic outputs of the RF, SVM and MLP models were then fused to create a new feature representation. This helps to avoid prediction bias and limits the leakage of information from the base to the meta-learning phase.

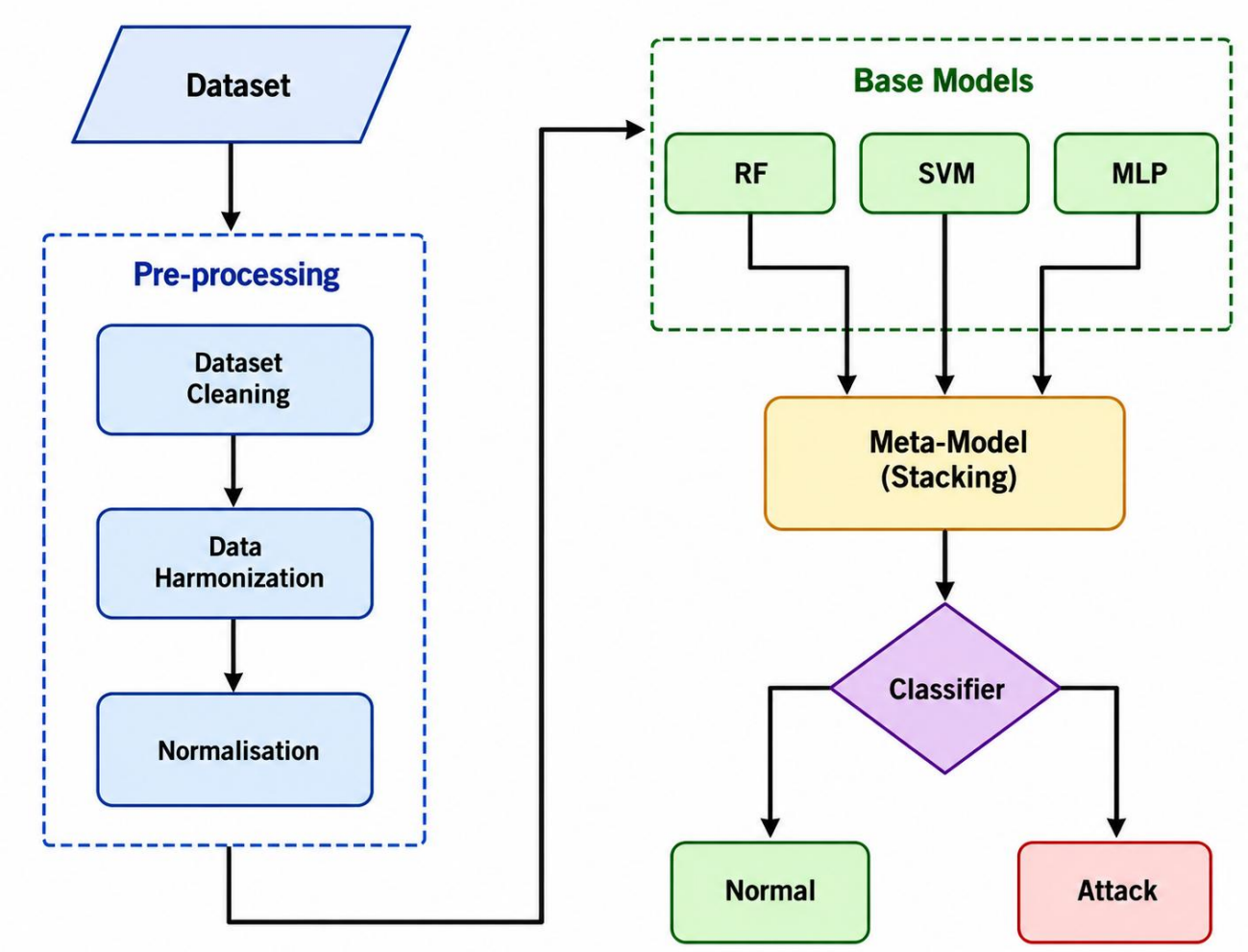

Fig 3: Stacking Ensemble Architecture for CSRF and CRSF Attack Detection

Let $h_1(x)$, $h_2(x)$, ..., $h_n(x)$ denote the predictions generated by the n base learners for an input feature vector x. This is done by combining the outputs of the base learners to form a secondary feature vector:
$z = [h_1(x), h_2(x), ..., h_n(x)]$
where z refers to the meta-feature vector generated from the prediction of the base classifiers. The vector z is used as input to the second stage of the stacking architecture. The second stage uses XGBoost as a meta-learner. The XGBoost model is trained using the meta-feature vector z generated from the out-of-fold predictions of the base learners. The final classification function is given as:

$H(x) = f(h_1(x), h_2(x), ..., h_n(x))$

Where

H(x) denotes the final classification output,

$f(\cdot)$ represents the XGBoost learning function,
and $h_i(x)$ represents the prediction generated by the i-th base learner.

To obtain the optimal meta-learner configuration, Randomized Search Cross-Validation was employed. A total of 20 sets of hyperparameters were randomly selected and tested using a 3-fold cross-validation. The mean ROC-AUC score across the validation folds was used as the performance metric. The best parameter setting, determined by the highest ROC-AUC, was then used to train the final XGBoost model on the full meta-training data. The optimization procedure is given in Algorithm 1.

Algorithm 1: XGBoost Meta-Learner Optimization

Input: X_meta_train, y_meta_train

Output: Optimized Meta-Learner M*

1: Define hyperparameter search space P

2: Initialize XGBoost model M

3: Best_AUC ← 0

4: Best_Params ← ∅

5: for i = 1 to 20 do

6: Randomly select parameter set p_i from P

7: Configure M using p_i

8: Perform 3-fold cross-validation

9: Compute mean ROC-AUC score AUC_i

10: if AUC_i > Best_AUC then

11: Best_AUC ← AUC_i

12: Best_Params ← p_i

13: end if

14: end for

15: Train final model M* using Best_Params

16: Return M*

The optimized XGBoost model serves as the meta-learner responsible for learning the relationships among the predictions generated by the RF, SVM, and MLP classifiers. The stacking framework takes advantage of the complementary strength of these different learning algorithms, thus boosting the attack detection capability and the generalization performance for XSS and CSRF attack classification.

Finally, the base learners' output is combined by the meta-learner to make the final classification decision. As shown in Fig.3, every Web request is classified as either Normal or Attack, which gives a common ground for detecting Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) attacks. The whole architecture was built on scikit-learn's Stacking Classifier, with XGBoost being the last estimator. The summary of the proposed stacking ensemble model is given in Algorithm 2.

Algorithm 2: Stacking Ensemble Detection Model
Input: Dataset, Training set, Testing set

Output: Trained model with performance metrics

1. Data Preprocessing:

a. Data cleaning

b. Data Merging and Feature Alignment

Model Training:

a. Perform a train–test split on the dataset

b. Train each base learner {RF, SVM, MLP } on Training Dataset

c. Train meta-leaner {XGboost} on the out-of-fold predictions of the base models

Evaluation:

a. Test each model on Testing Dataset using metrics:

-Accuracy, Precision, Recall, F1-Score and ROC-AUC

b. Compare results.

End Algorithm

3.4 Model Evaluation

The effectiveness of the proposed ensemble-based model was assessed by evaluating its performance to detect Cross-Site Scripting and Cross-Site Request Forgery attacks. The evaluation was based on the model's performance in correctly identifying legitimate and malicious HTTP requests and correctly categorising requests that belong to XSS and CSRF attacks.

The model was evaluated with commonly used classification performance metrics such as accuracy, precision, recall, F1-score and Receiver Operating Characteristic Area Under the Curve (ROC-AUC). These metrics were used to assess the overall classification performance and effectiveness of the proposed ensemble-based approach in detecting XSS and CSRF attacks.

Accuracy: It measures the proportion of instances that the classifier assigns to their correct classes out of the entire set of predictions [36]. In the case of the proposed XSS and CSRF detection model, a higher accuracy means that the model performs well in general, in terms of correctly detecting benign and malicious requests. It is determined by:

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (1)$$

Where TP, TN, FP and FN represent True Positive, True Negative, False Positive, and False Negative, respectively

Precision: Precision represents the ratio of the number of correctly classified positive cases to the total number of cases classified as positive [37]. For the purposes of XSS and CSRF detection, it refers to how accurate the model's predictions are for malicious requests. A high precision value means that the model makes fewer false alarms; that is, fewer benign requests are classified as malicious. It is determined by:

$$\text{Precision} = \frac{TP}{(TP + FP)} \quad (2)$$

Where TP and FP are True Positive and False Positive, respectively

Recall: Recall is the ratio of the number of correct positive instances to the total number of positive instances [37]. This is because it is a measure of the model's capacity to identify malicious HTTP requests, which can be a critical security concern in cybersecurity applications. A high recall value means that the model can detect more XSS and CSRF attacks with fewer false negatives. It is determined by the following:

$$\text{Recall} = \frac{TP}{(TP + FN)} \quad (3)$$

Where TP refers to True Positive, and FN refers to False Negative

F1-Score: F1-Score is a balanced measure of the classification performance, which is based on the combination of precision and recall [37]. It is the harmonic average of precision and recall and can be used to assess the model when it is important to also reduce false positives, in addition to accurately detecting malicious requests. F1 score is defined as:

$$\text{F1-Score} = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \quad (4)$$

Receiver Operating Characteristic Area Under the Curve (ROC-AUC): It is a typical measure of the quality of classification models, especially in binary classification models that have two classes. The ROC curve balances the True Positive Rate (TPR) and False Positive Rate (FPR) at different decision thresholds and the area under the curve or AUC is a single scalar statistic summarizing the discrimination ability of the model.

$$TPR = TP / (TP + FN) \quad (5)$$

$$FPR = FP / (FP + TN) \quad (6)$$

In practical implementations, AUC is often computed using the trapezoidal rule:

$$AUC = \Sigma_{i=1}^{n-1} (FPR_{i+1} - FPR_i) \times (TPR_{i+1} + TPR_i) / 2 \quad (7)$$

Where TP stands for True Positives, FP stands for False Positives, TN stands for True Negatives, FN stands for False Negatives, TPR stands for True Positive Rate and FPR stands for False Positive Rate.

4. RESULTS AND DISCUSSIONS

The performance of the stacking ensemble framework was evaluated using standard classification metrics. The metrics were applied to evaluate the overall classification and the model performance for distinguishing malicious and benign web requests at different decision thresholds.

4.1 Comparative Classification Performance

The comparative results indicate that there are significant differences between the performance of the classifiers evaluated. Random Forest model has an accuracy of 90%, precision of 0.84, recall of 0.88, and F1 score of 0.86. SVM achieved an accuracy of 91% and precision of 0.85, recall of 0.90 and F1-score of 0.87. The results show that both models were able to detect malicious web requests, but their performance was not as high as the neural and ensemble models.

The MLP obtained 95% accuracy and precision, recall, and F1-score were 0.93, 0.92, and 0.93, respectively. This shows that the MLP can learn complex relationships in the structured web-request features. The stacking ensemble, however, obtained the same accuracy of 95% and higher precision of 0.95 and the highest ROC-AUC value of 0.9808. This means that the ensemble was able to discriminate between benign and malicious requests with a higher accuracy than the MLP, while maintaining the same overall accuracy.

The accuracy of the ensemble (0.95) is especially significant in the context of web-attack detection, where false positives may result in legitimate requests being misidentified as attacks. The increased precision therefore indicates that the ensemble is capable of detecting attacks with a high degree of accuracy while minimizing false alarms of legitimate traffic as attacks. The full comparison is given in Table 1

Table 1: Comparison of the Models

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
RF	90%	0.84	0.88	0.86	0.9609
SVM	91%	0.85	0.90	0.87	0.9680
MLP	95%	0.93	0.92	0.93	0.9740
Ensemble	95%	0.95	0.91	0.93	0.9808

4.2 ROC-AUC Analysis

The ROC-AUC results are also evidence of the discriminative power of the models. Random Forest achieved an AUC of 0.9609, SVM 0.9680 and MLP 0.9740. The proposed ensemble got the maximum value of 0.9808.

The ROC-AUC values are shown to be improving steadily from Random Forest to SVM to the proposed ensemble, as presented in Figure 4. The difference between the ensemble and Random Forest is 0.0199, and the ensemble is better than SVM and MLP by 0.0128 and 0.0068, respectively. These differences are small but significant in terms of separation between benign and malicious requests for different classification thresholds

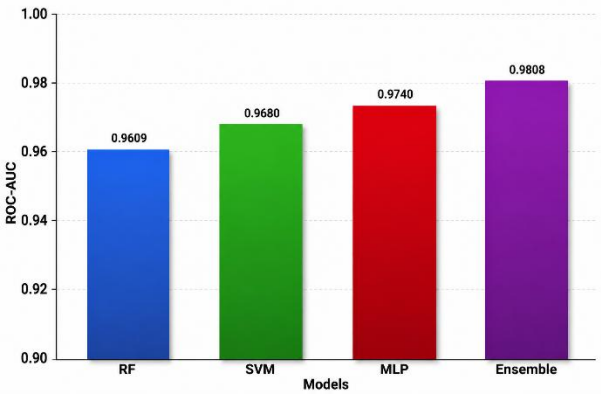

Fig 4 ROC-AUC Comparison Across Models

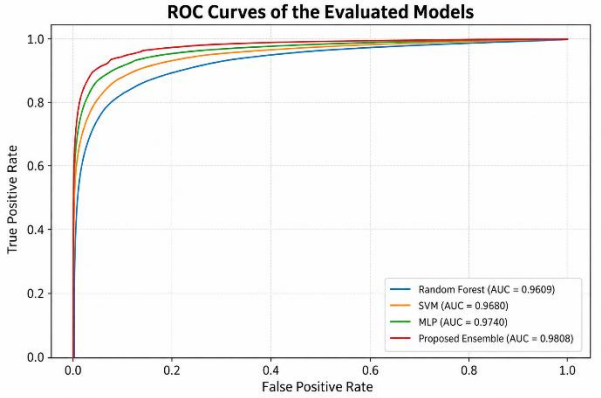

Fig 5 ROC Curve Across Models

This is also illustrated by the ROC curves in Figure 5. The proposed ensemble produces a curve that is closer to the upper left corner of the ROC space for most of the operating range, which represents a good compromise between the true-positive rate and the false-positive rate. This is crucial for security applications, where a detection system should be able to detect a high percentage of attacks, but with a low number of false alarms.

4.3 Performance Across XSS and CSRF Attack Sources

Further analysis of the source-specific was carried out to check the consistency of the proposed approach in both attack categories. The results are shown in Figure 6.

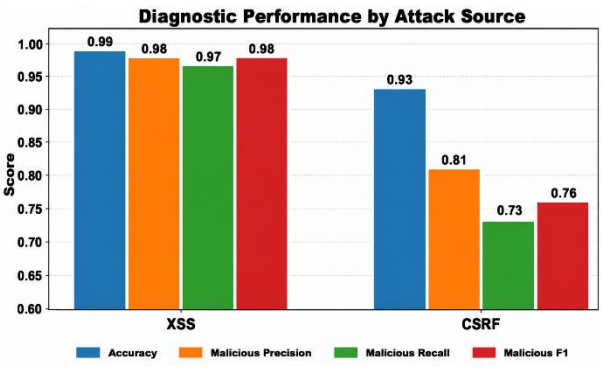

Fig 6 Performance Across XSS and CSRF Attack Sources

The system had an accuracy of around 99%, malicious precision of 98%, malicious recall of 97% and malicious F1 score of 98% for XSS traffic. The results show that the detection capability of XSS requests is high, and the extracted web-request features are helpful to distinguish XSS traffic from normal requests.

The performance for CSRF traffic was comparatively lower with around 93% accuracy, 81% malicious precision, 73% malicious recall and 76% malicious F1-score. The difference indicates that CSRF requests are harder to tell apart based on the available structured request features. While XSS attacks can often feature relatively unique patterns that are related to script, CSRF attacks can be very similar to legitimate authenticated requests. This means that some malicious CSRF requests might be harder to differentiate from legitimate requests.

However, the source-specific results show that the proposed model can detect both attack categories in a single framework without being limited to a particular web vulnerability. The results also indicate that CSRF detection is a significant field that needs further enhancement.

4.4 Overall Discussion

The results as a whole show that the stacking architecture yields competitive and, in most cases, better performance than the individual base learners. The ensemble consists of Random Forest, SVM, and MLP, with XGBoost as the meta-learner, enabling the integration of information from various learning methods to support the final classification result.

The results are especially remarkable because the ensemble and MLP had the same accuracy, but the ensemble had better precision and ROC-AUC values. This means that accuracy alone would not be sufficient to assess the difference between the classifiers. Therefore, the additional analyses on the attack-source were used to provide a more comprehensive evaluation of the proposed approach.

In general, the experimental results show that stacking is suitable for detecting unified XSS and CSRF. The lower CSRF recall found in the source-specific analysis, however, suggests that there is a need for further research to enhance the detection of challenging CSRF patterns and to test the model with larger and more varied real-world web-traffic datasets.

5. CONCLUSION AND FUTURE WORK

This study proposed a stacking ensemble learning framework for the detection of Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) attacks. This framework uses Random Forest, Multi-Layer Perceptron, and Support Vector Machine as base learners and XGBoost as a meta-learner to boost the prediction performance. Experimental results demonstrated that the proposed ensemble model outperformed individual classifiers in terms of ROC-AUC, precision, and overall robustness. The MLP had similar accuracy, but the ensemble model demonstrated better discriminative power with its ROC AUC score of 0.9808. It proves that stacking can be an effective technique for enhancing the reliability of classification in web security applications.

Future research should focus on the use of larger and more diverse data sets to enhance the generalisation capabilities of the detection models in various Web environments. Using traffic of various web applications and different operating environments, researchers will be able to make sure that models can change appropriately with the changing trends of XSS and

CSRF attacks. Further, based on this, the real-time application of the proposed ensemble system should also be worked towards. Even though design, simulation, and evaluation were the main priorities in this study, the implementation of the model into a live setting would provide a better understanding of its responsiveness to real-world traffic and its ability to counteract active threats during their occurrence.

6. REFERENCES

[1] Dwivedi, Y., Williams, M., Mitra, A., Niranjan, S., and Weerakkody, V. (2011). Understanding advances in web technologies: evolution from web 2.0 to web 3.0.

[2] Xia, L., Baghaie, S., and Sajadi, S. M. (2024). The digital economy: Challenges and opportunities in the new era of technology and electronic communications. *Ain Shams Engineering Journal*, *15*(2), 102411.

[3] Diwan, T. D. (2021). An investigation and analysis of cyber security information systems: latest trends and future suggestion. *Information Technology in Industry*, 9(2), 477-492.

[4] OWASP Foundation. (2025a). *OWASP Top 10: 2025 – A05: Injection.* Retrieve from *https://owasp.org/Top10/2025/A05_2025-Injection/* on July 30.2025

[5] OWASP Foundation. (2025b). *OWASP Top 10: 2025 – A01: Broken Access Control*. Retrieve from https://owasp.org/Top10/2025/A01_2025-Broken_Access_Control/ on July 30.2025

[6] Rodriguez, G. E., Torres, J. G., Flores, P., and Benavides, D. E. (2020). Cross-site scripting (XSS) attacks and mitigation: *A survey. Computer Networks*, 166, 106960.

[7] Alaoui, R. L., and Nfaoui, E. H. (2022). Deep learning for vulnerability and attack detection on web applications: A systematic literature review. *Future Internet,* 14(4), 118.

[8] Razzaq, A., Latif, K., Ahmad, H. F., Hur, A., Anwar, Z., and Bloodsworth, P. C. (2014). Semantic security against web application attacks. *Information Sciences*, *254*, 19-38.

[9] Ali, A. H., Charfeddine, M., Ammar, B., Hamed, B. B., Albalwy, F., Alqarafi, A., & Hussain, A. (2024). Unveiling machine learning strategies and considerations in intrusion detection systems: a comprehensive survey. *Frontiers in Computer Science*, *6*, 1387354.

[10] Momand, A., Jan, S. U., & Ramzan, N. (2023). A systematic and comprehensive survey of recent advances in intrusion detection systems using machine learning: Deep learning, datasets, and attack taxonomy. *Journal of sensors*, *2023*(1), 6048087.

[11] Chua, T. H., & Salam, I. (2022). Evaluation of machine learning algorithms in network-based intrusion detectionsystem. *arXiv preprint arXiv:2203.05232*.

[12] Pawlicki, M., Kozik, R., & Choraś, M. (2022). A survey on neural networks for (cyber-) security and (cyber-) security of neural networks. *Neurocomputing*, *500*, 1075-1087.

[13] Hannousse, A., Yahiouche, S., and Nait-Hamoud, M. C. (2024). Twenty-two years since revealing cross-site scripting attacks: A systematic mapping and a comprehensive survey. *Computer Science Review*, *52*, 100634.

[14] Gupta, S., and Gupta, B. B. (2017). Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. *International Journal of System Assurance Engineering and Management*, *8*(Suppl 1), 512-530

[15] Luo, A., Huang, W. and Fan, A.W. (2019). CNN-based approach to the detection of SQL injection attacks. *In 2019 IEEE/ACIS 18th International Conference on Computer and Information Science* (ICIS). 320-324.

[16] Herman, H., Riadi, I., and Kurniawan, Y., (2023). Vulnerability detection with K-nearest neighbor and naive Bayes method using machine learning. *Int. J. Artif. Intell*. Res. 7, 1

[17] Rankothge, W. H., & Randeniya, S. M. N. (2020). Identification and mitigation tool for cross-site request forgery (CSRF). In *2020 IEEE 8th R10 Humanitarian Technology Conference (R10-HTC)* (pp. 1–5). IEEE. https://doi.org/10.1109/R10-HTC49770.2020.9357029

[18] Bohara, R., Arjun, V. V. J., Jaiswal, D. J., Nikhil, M., Geetha, G., Pandey, B., and Raghav, U. R. (2023). A survey paper on cross-site scripting (XSS). In *Proceedings of the International Conference on Innovative Computing and Communication (ICICC) 2022* (pp. 1–5). SSRN.

[19] Abaimov, S., and Bianchi G (2021). A survey on the application of deep learning for code injection detection. *Array. 11(June)*:100077.

[20] Ali, A. H., Charfeddine, M., Ammar, B., Hamed, B. B., Albalwy, F., Alqarafi, A., & Hussain, A. (2024). Unveiling machine learning strategies and considerations in intrusion detection systems: a comprehensive survey. *Frontiers in Computer Science*, *6*, 1387354.

[21] Mokbal, F. M. M., Dan, W., Imran, A., Jiuchuan, L., Akhtar, F., and Xiaoxi, W. (2019). MLPXSS: an integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique. *IEEE Access*, 7, 100567-100580

[22] Stency, V. S., and Mohanasundaram, N. (2021). A study on XSS attacks: Intelligent detection methods. *Journal of Physics: Conference Series, 1767*(1), 012047. IOP Publishing

[23] Mokbal, F.M.M., Dan, W., Xiaoxi, W., Wenbin, Z., and Lihua, F (2021). XGBXSS: An Extreme Gradient Boosting Detection Framework for Cross-Site Scripting Attacks Based on Hybrid Feature Selection Approach and Parameters Optimization. *J. Inf. Secur. Appl*. 2021,58, 102813.

[24] Kumar J, Santhanavijayan A, Rajendran B. (2022) Cross site scripting attacks classification using convolutional neural network. *IEEE ICCCI*; 2022:1-6.

[25] Alaoui, R.L. (2022). Web attacks detection using stacked generalization ensemble for LSTMs and word embedding. *Procedia Computer Science, 215*, 687–696.

[26] Tadhani, J. R., Vekariya, V., Sorathiya, V., Alshathri, S., and El-Shafai, W. (2024). Securing web applications against XSS and SQLi attacks using a novel deep learning approach. *Scientific Reports*, 14(1), 1803.

[27] Muralitharan, R., Torres, J. G., Flores and Alshammari, M. (2024). Hybrid stacking ensemble deep neural network (DNN) model to detect XSS, combining traditional ML classifiers. *Information*, 15(7), 424.

[28] Alhamyani, R., and Alshammari, M. (2024). Machine learning-driven detection of cross-site scripting attacks. *Information,* 15(7), 420.

[29] Nagpal, B., Chauhan, N., and Singh, N. (2017). SECSIX: security engine for CSRF, SQL injection and XSS attacks. *International Journal of System Assurance Engineering and Management,* 8, 631-644.

[30] Pellegrino, G., Johns, M., Koch, S., Backes, M., and Rossow, C. (2017). Deemon: Detecting CSRF with dynamic analysis and property graphs. *In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1757-1771).

[31] Calzavara. S, M. Conti, R. Focardi, A. Rabitti, and G. Tolomei, (2019) "Mitch: A Machine Learning Approach to the Black-Box Detection of CSRF Vulnerabilities," *in 2019 IEEE European Symposium on Security and Privacy (EuroSandP), Stockholm, Sweden*, pp. 528–543.

[32] Hadavi, M. A., and Sadeghi, S. (2021). Automatic black-box detection of resistance against CSRF vulnerabilities in web applications. *Journal of Computer Science, 8*(1).

[33] Ismail, M. A., and Hassan, M. M. (2021). An automated detection system of cross site request forgery (CSRF) vulnerability in web applications. *International Journal of Innovative Science and Research Technology*, *6*(10), 582–586

[34] Sravani, N., Raju, O. S., Harish, C., Kumar, B. A., and Anirudh (2024), S. Machine Learning for Web Vulnerability Detection: The Case of Cross-Site Request Forgery. *International Journal of Information Technology and Computer Engineering, 12*(1), 162–171.

[35] Kshetri, N., Kumar, D., Hutson, J., Kaur, N., and Osama, O. F. (2024). algoXSSF: Detection and analysis of cross-site request forgery (XSRF) and cross-site scripting (XSS) attacks via Machine learning algorithms. *In 2024 12th International Symposium on Digital Forensics and Security* (ISDFS) (pp. 1-8). IEEE.

[36] Liu, Y., Zhou, Y., Wen, S., and Tang, C. (2014). A strategy on selecting performance metrics for classifier evaluation. *International Journal of Mobile Computing and Multimedia Communications, 6*(4), 20–35. https://doi.org/10.4018/IJMCMC.2014100102

[37] Tasnim, A., Saiduzzaman, M., Rahman, M. A., Akhter, J., and Rahaman, A. S. M. M. (2022). Performance evaluation of multiple classifiers for predicting fake news. Journal of Computer and Communications, 10(9), 1–21. https://doi.org/10.4236/jcc.2022.109001.