Library Progress International *Print version ISSN 0970 1052*
Vol.44 No. 3, July-December 2024: P.24245-24254 *Online version ISSN 2320 317X*

Original Article *Available online at www.bpasjournals.com*

Detecting Cross-Site Scripting (XSS) Using Machine Learning & Deep Learning

Nilesh Patil[1*], Chinmay Raut[1*], Karan Jain[1†], Krish Gogri[1†], Deep Gada[1†], Hitansh Shah[1†]

[1]Computer Engineering, Dwarkadas J. Sanghvi College of Engineering, Bhaktivedanta Swami Rd., Mumbai, 400056, Maharashtra, India.
[*]Corresponding author(s). E-mail(s): nilesh.p@djsce.ac.in, chinmay.raut@djsce.ac.in;.

How to cite this article: Nilesh Patil, Chinmay Raut, Karan Jain, Krish Gogri, Deep Gada, Hitansh Shah (2024). Detecting Cross-Site Scripting (XSS) Using Machine Learning & Deep Learning. *Library Progress International,* 44(3), 24245-24254

Abstract
Cross-Site Scripting(XSS) is a critical vulnerability in web applications, in which attackers inject malicious scripts to compromise user data. In this paper, we highlight how cross-site scripting can be detected using various machine learning models like SVM, Logistic Regression, Random Forest, Naive Bayes, Gradient Boosting and deep learning LSTM model. Our findings show that the proposed models significantly enhance the accuracy of detecting malicious scripts, achiev-ing an accuracy of 89%. These methods provide an enhanced security to protect user data by detecting malicious scripts. Our findings offer insights in deploying a layered security approach against cross-site scripting for real world applica-tions. Future work will focus on using hybrid models for better accuracy and also more variety of datasets. Reinforcement learning could be explored for detection capabilities.
Keywords: Cross-Site Scripting(XSS), Machine Learning(ML), Deep Learning(DL), Intrusion Detection Systems(IDS), Dimensionality Reduction, Hyperparameter Tuning

1 Introduction

Cross Site Scripting (XSS) attack is a significant threat to web applications where an attacker aims to execute malicious scripts in victim's web browser by injecting malicious code in the legitimate web application. The actual attack occurs when the victim visits the web page or web application that executes the malicious code. The web page or web application becomes a vehicle to deliver the malicious script to the user's browser. Traditional detection methods, while effective, often fail to keep up with the evolving complexity of XSS attacks. This research paper explores the application of machine learning techniques,to improve detection of XSS attacks.

In this study, we preprocess a dataset of containing a broad range of XSS attack pat-terns,therefore preparing it for machine learning. A deep learning model, comprising of Long Short-Term Memory(LSTM) and dense layers, is built using Keras, with hyper-parameters fine-tuned through Keras Tuner which allows the model to general-ize and reduce the risk of overfitting. Also, various accuracy metrics and classification reports are used to evaluate model's performance.

The aim of this paper is to provide a better solution for improving web security and also how with the help of various machine learning models we can enhance the detection of XSS attacks, making web applications more secure and reliable to use.

2 Literature Review

In [1], a model is proposed that can efficiently detect patterns and atypical behaviors associated with XSS attacks, thereby minimizing the likelihood of false alarms. How-ever, it is limited to blind XSS attacks and not very effective against DOM based or reflected XSS.

In [2], utilizing keyword recognition as the foundation of the theory to assess the velocity of an attack is done, with a primary emphasis on the celerity of detection rather than attack identification, thereby facilitating the attainment of an expeditious resolution. However, there is no method to prevent overfitting or bias.

In [3], a combination of Principal Component Analysis (PCA) and t-SNE (t-distributed Stochastic Neighbor Embedding) can be employed to optimize the visualization of high-dimensional data. This method addresses the potential compu-tational inefficiencies associated with t-SNE alone. By using PCA as a dimensionality reduction technique prior to applying t-SNE, we can significantly enhance processing speed while preserving the integrity of the data representation.

In [4], the model to enhance real time cross-site scripting (XSS) detection capabilities specially those who involve credit card frauds by extracting features from http requests to focuses more on analyzing specific characters that indicate malicious activity.

In [5], a method was suggested that uses principal component analysis and factor analysis for feature extraction this method focuses on identifying feature that showed trait of these vulnerabilities leading to enhance detection accuracy hence improving real time detection capabilities.

In [6], the uses of decision tree model were highlighted to enhance performance of detecting cross-site scripting (XSS). Implemented a model using Random Forest and ADTree to classify and identify XSS attacks effectively by extracting Social Networking Site Keywords, HTML and URLs hence showed how effective decision tree model are to typical machine learning approaches.

In [7], a model that utilizes lexical analysis of JavaScript tokens to identify obfus-cated JavaScript attacks was implemented. This method examines the structures and patterns within JavaScript tokens, discerning characteristics of obfuscation that could indicate malicious intent. Consequently, it enhances the detection of concealed threats. In [8], a model was deployed which uses lexical analysis of JavaScript token to detect obfuscated JavaScript attacks. This approach closely studied the structures and pat-terns in the JavaScript token and identified traits of obfuscation that may lead to malicious intent hence helping to detect more hidden threats.

In [9], a detection system was proposed that utilizes sensitive word replacement and encoding obfuscation techniques. This system aims to preserve the functionality of payloads while bypassing security filters and disguising malicious scripts. By doing so, it becomes more challenging for Intrusion Detection Systems (IDS) to differentiate between harmful and legitimate content.

In [10], a threat intelligence model was proposed that utilizes reinforcement learning (RL), statistical inference, and accuracy variability. Past historical attack data and historical trends analysis was used to train those models to make them effective against different types of attacks.

In [11], an approach that integrated signature-based pattern matching algorithm with reinforcement learning to detect cloud based crossed site scripting attack was proposed. Signature based pattern matching algorithm helped to detect any such know attack pattern helping RL model to adapt and learn effectively.

In [12], three deep learning architectures were deployed: Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), and Long Short-Term Memory networks (LSTM) to detect various cyber attacks, including Denial of Service (DoS), Cross-Site Scripting (XSS), and SQL injection attacks. These models

facilitated general pattern recognition and excelled in identifying spatial hierarchies in data, making them suit-able for recognizing patterns in network traffic. Additionally, they captured temporal dependencies, which are essential for comprehending sequences in attack data.

In [13], Long Short-Term Memory (LSTM) classifiers were implemented in conjunc-tion with Principal Component Analysis (PCA) and Mutual Information (MI) for the purpose of detecting cross-site scripting attacks. It demonstrates that dimen-sionality reduction techniques enhances the detection of multi-class attacks. Further research into real-time applications and improved detection strategies for obfuscated code should be employed, with the ultimate goal of developing more effective defense mechanisms against cyber threats.

In [14], a model was deployed to emphasis more on feature extraction from HTTP request this was accomplish by deploying three machine learning models J48, OneR, and Na¨ıve Bayes—to detect XSS vulnerabilities in web applications. This method aimed to classify inputs as either benign or malicious focusing on lower layer data but neglected higher layer of web architecture.

3 Methodology

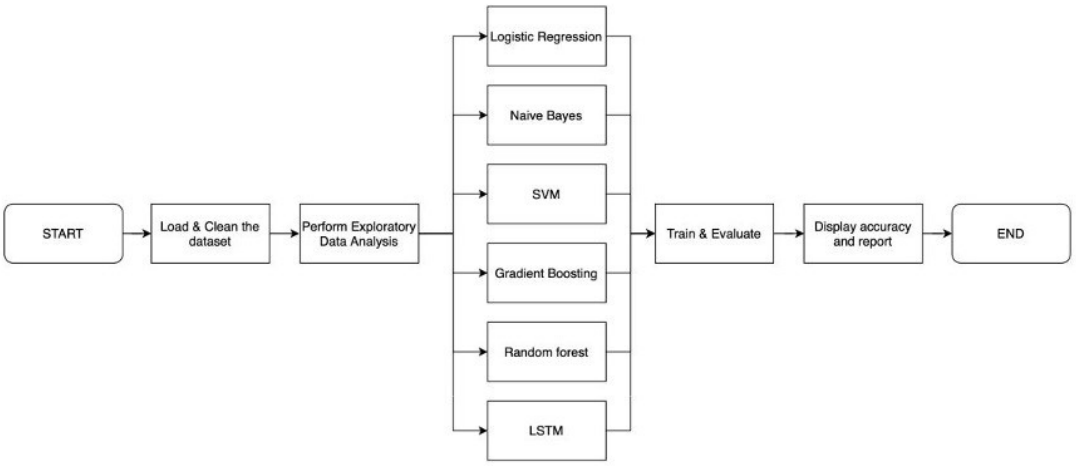

Fig. 1 Workflow of Machine Learning Model Selection and Evaluation

Logistic regression

A statistical model that can predict the probability of an event based on prior observation giving output in a binary format (0 or1) hence predicting the probability that a given input belongs to a particular category. The logistic regression model is represented as follows:

$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_k X_k)}} \qquad (1)$$

where,

- P(Y=1|X) is the probability that the dependent variable Y equals 1 given the independent variables X.
- e is the base of the natural logarithm.
- 0 is the intercept (constant term).
- 1,2,...,k are the coefficients for each independent X1,X2,...,Xk

Naive Bayes

An algorithm used for classification task by assuming that features used for clas-sification are conditionally independent given class label thus when a class label is known then ones feature does not affect the presence

of another feature. The Bayes theorem formula used for probabilistic analysis is given by:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \quad (2)$$

Where:

- P(A|B) is the posterior probability, the probability of class A given feature B.
- P(B|A) is the likelihood, the probability of feature B given class A.
- P(A) is the prior probability of class A.
- P(B) is the evidence, the total probability of feature B.

SVM

Support Vector Machine (SVM) is a machine learning algorithm primarily used for classification tasks by finding the optimal hyperplane that best separates different classes in the feature space. Hence it is a decision boundary that maximizes the margin between the closest data points from each class, known as support vectors helping the model to enhance its generalisation capability. For the SVM model, the classification constraint is:

$$y_i(w \cdot x_i + b) \geq 1 \quad \text{for all } i \quad (3)$$

where:

- w is the weight vector (normal to the hyperplane),
- xi are the input feature vectors,
- yi are the class labels (+1 or -1),
- b is the bias term.

Gradient boosting

A ensemble learning method which helps a model to correct errors made by other models hence builds a predictive model in a stage wise manner by combining multiple weak model hence eliminating a loss function by fitting new model in the residual of existing model allowing it to be highly efficient for complex and large dataset.

$F_m(x) = F_{m-1}(x) + \nu h_m(x)$ (4) where:

- $F_m(x)$ is the updated model after adding the m-th weak learner.
- $F_{m-1}(x)$ is the previous model.
- ν is the learning rate (a value between 0 and 1).
- $h_m(x)$ is the new weak learner fitted to the pseudo-residuals.

Random Forest

A machine learning algorithm that employs an assembly of decision tree where individual tree is trained on a random subset of data and feature, prediction from each decision tree is aggregated on the bases of majority voting for classification or averaging for regression for the final prediction

$$\text{Predicted Class} = \text{mode}(T_1(x), T_2(x), \ldots, T_n(x)) \quad (5)$$

$$\text{Predicted Value} = \frac{1}{n} \sum_{i=1}^{n} T_i(x) \quad (6)$$

where:

- n is the total number of trees in the forest.
- Ti(x) is the prediction made by the i-th tree for input x.

LSTM

A type of recurrent neural network that learns from subsequences of data to elim-inate challenges related to long term dependencies thus remembering information for extended period of time

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (7)$$

This gate outputs values between 0 and 1, indicating how much of the previous cell state should be retained.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (8)$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (9)$$

The first equation decides which values to update, while the second generates candidate values for updating the cell state.

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \quad (10)$$

This combines the retained information from the previous state with new candidate values.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (11)$$

$$h_t = o_t * \tanh(c_t) \quad (12)$$

The first equation determines what part of the cell state will be output, and the second produces the hidden state. Where:

- xt is the input at time step t.
- ht−1 is the hidden state from the previous time step.
- Wf , Wi, Wc, Wo are weight matrices for each gate.
- bf , bi, bc, bo are bias vectors for each gate.
- σ is the sigmoid activation function.

4 Experimentation and Results

1.Experimental Setup

The experiment has been conducted to detect Cross-Site Scripting(XSS) attacks using supervised ML based classification algorithms and DL based neural networks. The model architecture implements both ML and DL techniques separately on kaggle dataset link consisting of all kinds scripts (malicious and non-malicious) to detect whether a system is vulnerable or not.

Dataset:-XSS Detection Dataset was split into 80% training data and 20% test data. Then data preprocessing is applied on script feature to remove HTML,XML tags and reduce unnecessary whitespaces.

2. Evaluation Metrics

The performance of a classification model is evaluated using metrics such as
Accuracy, Precision, Recall, and F1 Score. Below are the equations for these metrics:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad (13)$$

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{14}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{15}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{16}$$

These metrics provide a comprehensive evaluation of the model's performance by balancing correctness, positive prediction quality, and the ability to find all positive instances.

3.Results

The findings demonstrated in the table below emphasise the effectiveness of using various machine learning and deep learning techniques for detecting Cross-Site Script-ing(XSS) attacks, representing a remarkable enhancement in web security. These results depict how different machine learning and deep learning approaches can lead to more robust defences against XSS vulnerabilities, improving the overall security of web applications. Moreover, the proposed detection method has gone through rigor-ous testing and has proved to be more accurate than a lot of existing systems. This comparative analysis assesses its performance and highlights the strengths and weak-nesses of the proposed method relative to established approaches. By analysing these metrics, we gain valuable insights into the efficacy of the new method, ensuring that it is both innovative and practical for real-world applications.

These models were run in the Google Colab environment, which helped ensure consistency and better accuracy. Hyperparameter tuning was used to get accurate results and better performance.

Model	Precision	Recall	F1 Score	Accuracy
Logistic Regression	0.91	0.89	0.89	89.41%
Naive Bayes	0.84	0.74	0.73	74.87%
Random Forest	0.91	0.89	0.89	89.37%
SVM	0.91	0.90	0.89	89.26%
Gradient Boosting	0.90	0.89	0.89	89.11%

Table 1 Performance Metrics of Different Models

From the table, we can see that Logistic regression showed better performance than all other algorithms, having an accuracy of 89.41%. But, others like Random forest, SVM, Gradient Boosting, also showed good perfor-mance having an accuracy of around 89%. Naive Bayes showed poor performance with accuracy of just 74.87%, which meant that for detecting cross site scripting naive bayes is not a suitable method.

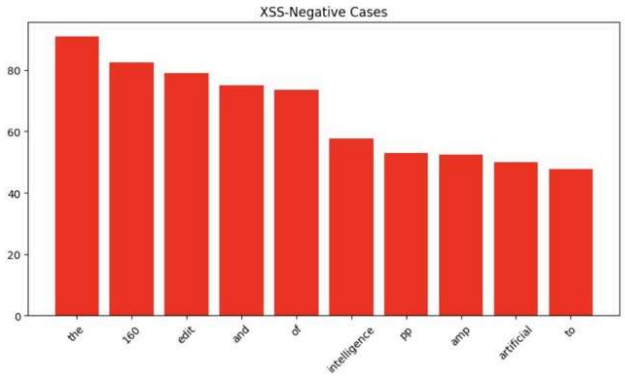

Fig. 2

The red bar chart in Fig.2 shows that XSS attacks were not detected in cases with words or tokens like 160, 10, edit, and , of, intelligence, etc. These words usually appear in benign sites or non-malicious sites. This indicates that sites having these keywords are generally safe and being used in everyday and legitimate content.

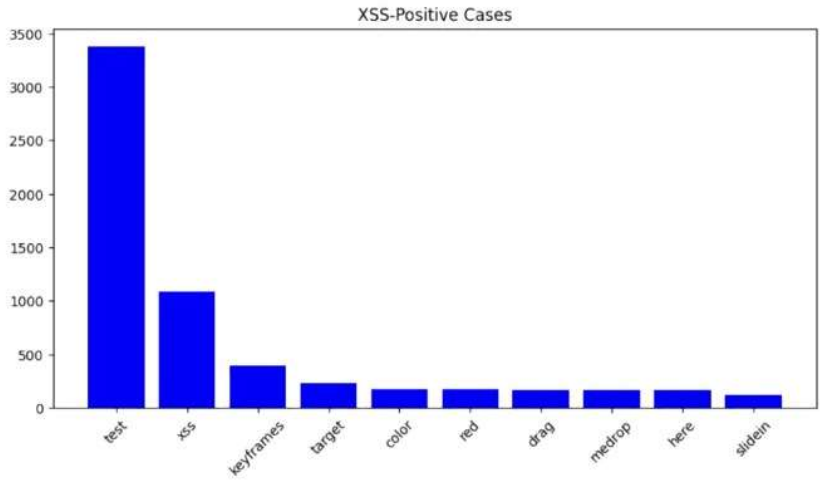

Fig. 3

The blue bar chart in Fig.3 shows that XSS attacks were detected in cases with words or tokens like test, xss, keyframes, target, and color, with test being the most frequently used. This indicates that sites having these keywords might be used for code testing purposes or they can be actually used in malicious scripts.

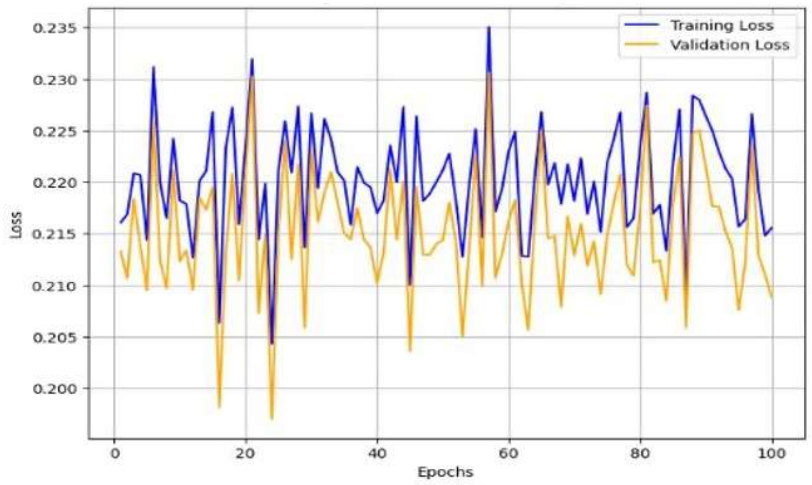

Fig. 4 Training and Validation Loss

The graph in Fig.4 shows the training loss(blue) and validation loss(orange) over 100 epochs. The validation loss remains lower than the training loss, which indicates that the model is not over-fitting and has generalized well to unseen data. Also, the losses decline smoothly showing stable learning.

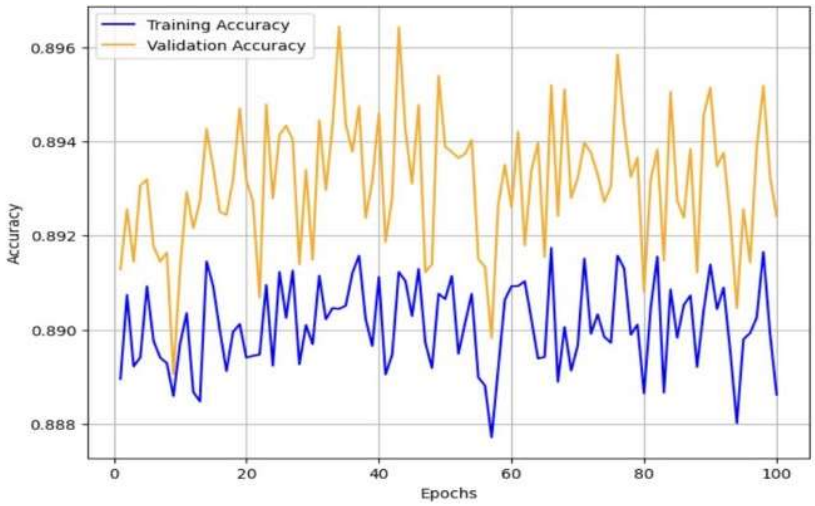

Fig. 5 Training and Validation Accuracy

The graph in Fig.5 shows the training accuracy(blue) and validation accu-racy(orange) over 100 epochs. The validation accuracy is seen higher than the training accuracy, which indicates the model is generalizing well and not overfitting. Both accuracies are remaining stable,suggesting that the model is learning well.

5 Conclusion

Thus, By implementing ML and DL techniques,we have achieved a decent aver-age accuracy of 89%.This accuracy has helped us greatly in detecting malicious HTML,XML scripts that can prevent rendering of scripts in the victim's browser and prevent system compromise as well. Various hyperparameters in case of ML and epochs in case of DL have been applied to enhance accuracy and efficiency of model. More-over,the dataset used in the experiment is quite fairly unbiased dataset to avoid any kind of skewed results. Hence, both approaches demonstrates their robustness in iden-tifying potentially harmful code injections, suggesting that these models can serve as reliable tools for enhancing web security and prevent cyber attacks based on XSS.

6 Future Scope

Future research could improve this study by using hybrid models that combine ML and DL for better detection accuracy and efficiency. By using various feature engineering techniques and hyperparameter tuning could also help improve model performance. Also, considering real-time detection frameworks with larger and more wide variety of datasets could further make this model useful in many more environments. Lastly, by deploying these models in real-world web applications we could get valuable insights for future security measures against XSS attacks.

References

[1] Kaur, G., al.: Detection of blind and stored xss attacks. Security Journal 15, 130–142 (2018). Focuses on blind and stored XSS attacks. Lacks methods for DOM-based or reflected XSS.

[2] Wang, R., al.: Keyword-based xss detection on social networking sites. Journal of Web Security 7, 234–245 (2015). Detection focused on speed and keyword matching, limited to social networks. Overfitting prevention not addressed.

[3] Kascheev, M., Olenchikova, S.: Comparative study of machine learning models for xss detection. Computing and Security 16, 345–360 (2020). Evaluates decision tree, Na¨ıve Bayes, logistic regression, and SVM models for XSS detection. PCA used to optimize t-SNE performance.

[4] Alam, A., Pachauri, S.: Credit card fraud and xss detection using feature extrac-tion from http requests. International Journal of Security and Networks (2017). Features extracted from HTTP requests to improve accuracy. Could include JavaScript keywords, HTML tags, malicious URLs, and IP addresses for improved real-time detection.

[5] Munonye, T., P´eter, I.: Feature analysis for xss and oauth vulnerability detection. Journal of Web Vulnerabilities 13, 183–195 (2021). PCA and Factor analysis for detecting XSS and OAuth vulnerabilities. JavaScript feature extraction for real-time results suggested.

[6] Rathore, S., al.: Social networking site attack detection using decision trees. Cybersecurity Studies 8, 154–169 (2017). Decision tree models, specifically Ran-dom Forest and ADTree, for XSS detection. Preventing overfitting is suggested.

[7] Khan, M., al.: Detection of obfuscated javascript attacks using lexical analysis. Journal of Secure Web Systems 6, 78–92 (2017). Lexical analysis of JavaScript tokens for detecting obfuscated attacks. Static analysis may be slow for large datasets.

[8] Hoang, X.: Xss detection using pca and cart on web logs. Cyber Systems Journal 14, 56–68 (2020). Uses PCA and CART algorithms for detecting XSS on web logs. Further obfuscation detection recommended.

[9] Fang, L., al.: Sensitive word replacement for xss evasion detection. Web Security Journal 10, 210–222 (2019). Applies word replacement and encoding obfuscation for bypassing security filters. Multiple datasets for testing advised.

[10] Tariq, A., al.: Reinforcement learning-based model for xss detection. Journal of Threat Intelligence 19, 118–132 (2021). Model based on reinforcement learn-ing and statistical inference for XSS attack detection. Patterns susceptible to alteration.

[11] Praise, A., al.: Cloud-based xss attack detection using reinforcement learning. Journal of Cybersecurity Research 12, 345–356 (2020). Signature-based pat-tern matching algorithm integrated with reinforcement learning. Lacks evasion detection strategy in firewall implementation.

[12] Feng, B., al.: Deep learning approaches for xss detection. International Journal of Cybersecurity 10, 120–134 (2019). DNN, CNN, and LSTM models used for XSS detection. Testing with updated datasets suggested to verify accuracy.

[13] Laghrissi, A., al.: Lstm-pca classification for multi-class xss and dos detec-tion. Cyber Intelligence Review 14, 78–89 (2021). Combines LSTM classifiers with PCA and MI for effective XSS and DOS classification. Further testing on obfuscated code needed.

[14] Sharma, S., al.: Machine learning models for http-based xss detection. Journal of Network Security 11, 98–110 (2020). Uses J48, OneR, and Na¨ıve Bayes algorithms for HTTP-based XSS detection. Limited to

lower-layer data.

[15] Abaimov, G., Bianchi, F.: Keyword-based preprocessing for xss detection. Journal of Secure Web Applications 9, 45–59 (2019). Employs a pre-processing module for attack detection using keywords and symbols. Extending detection for broader cyberattacks suggested.

[16] Li, Z., al.: Transformer-based detection of complex xss patterns. Journal of AI in Security 18, 102–118 (2022). Introduced transformer-based model for complex XSS patterns. Further real-world validation recommended.

[17] Zhang, T., al.: Cnn with attention mechanisms for large-scale xss detection. Machine Learning in Security 20, 210–225 (2023). Enhanced XSS detection com-bining CNN with attention for large-scale datasets. Optimization for real-time data recommended.

[18] Chauhan, R., al.: Federated learning in decentralized xss detection. Decentralized Systems Security 22, 56–73 (2023). Utilizes federated learning for XSS detection. Improved privacy-preserving techniques suggested.