

# XSSer: hybrid deep learning for enhanced cross-site scripting detection

Ammar Odeh, Anas Abu Taleb

Department of Computer Science, Princess Sumaya University of Technology, Amman, Jordan

## Article Info

### Article history:

Received Nov 20, 2023

Revised Feb 12, 2024

Accepted Mar 20, 2024

### Keywords:

Content security policies

Cross-site scripting

Cyberattack

Deep learning

Intrusion detection system

## ABSTRACT

The importance of an effective cross-site scripting (XSS) detection system cannot be overstated in web security. XSS attacks continue to be a prevalent and severe threat to web applications, making the need for robust detection systems more crucial than ever. This paper introduced a hybrid model that leverages deep learning algorithms, combining recurrent neural network (RNN) and convolutional neural network (CNN) architectures. Our hybrid RNN-CNN model emerged as the top performer in our evaluation, demonstrating outstanding performance across key metrics. It achieved an impressive accuracy of 96.74%, excelling in accurate predictions. Notably, the precision score reached an impressive 97.78%, highlighting its precision in identifying positive instances while minimizing false positives. Furthermore, the model's recall score of 95.65% showcased its ability to capture a substantial portion of true positive instances. This resulted in an exceptional F1-Score of 96.70, underlining the model's remarkable balance between precision and recall. Compared to other models in the evaluation, our proposed model unequivocally demonstrated its leadership, emphasizing its excellence in detecting potential XSS vulnerabilities within web content.

This is an open access article under the [CC BY-SA](#) license.



## Corresponding Author:

Ammar Odeh

Department of Computer Science, Princess Sumaya University of Technology

Amman 1196, Jordan

Email: a.odeh@psut.edu.jo

## 1. INTRODUCTION

Cross-site scripting (XSS) represents a security flaw wherein a web application permits users to embed harmful scripts into web pages seen by others. This widespread vulnerability among web applications can lead to severe consequences such as data theft and session hijacking [1], [2]. XSS attacks transpire when untrusted data is integrated into a web page without appropriate validation or encoding, enabling attackers to implant malevolent scripts (typically in JavaScript). Consequently, these scripts execute when other users access the page. Figure 1 delineates the three main categories of XSS attacks.

- Stored XSS (persistent XSS): in this attack, the injected script is permanently stored on the target server and served to users who access the affected page. For example, an attacker might post a malicious script on a forum, and anyone who views that forum post will execute the script [3]-[5].
- Reflected XSS: in a reflected XSS attack, the injected script is not stored on the target server but is instead reflected off a web application, often via a URL. When a user clicks on a specially crafted link, the script is executed in their browser [6], [7].
- DOM-based XSS: this is a more advanced form of XSS, where the attack occurs in the web page's document object model (DOM). The attacker manipulates the DOM in a way that causes the script to run when the user loads or controls the page [8], [9].

It's crucial to employ effective input validation and output encoding to reduce XSS vulnerabilities. This includes thoroughly validating and cleaning user inputs while encoding any output that contains user-generated content. Additionally, web developers ought to integrate security measures like content security policy (CSP) to hinder the execution of inline scripts [7], [8].

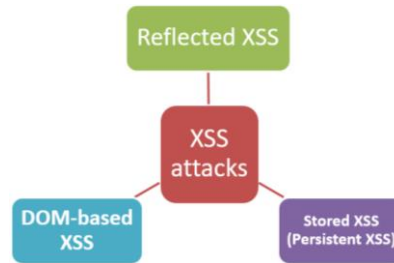


Figure 1. Types of XSS attacks

To protect against XSS attacks, web application developers and users should be aware of the risks and best practices for prevention. Users should be cautious about clicking on untrusted links and use browser extensions blocking known malicious scripts. Developers should follow secure coding practices, validate and sanitize user input, and use security libraries and frameworks to help prevent XSS vulnerabilities. Regular security testing, including automated and manual security audits, is essential to identify and fix potential XSS issues in web applications [10], [11].

XSS detection plays a pivotal role in contemporary cybersecurity, addressing a range of critical security concerns. XSS attacks can lead to data breaches, wherein sensitive user information, such as credentials and personal data, is compromised. Effectively detecting XSS attacks is fundamental in protecting this data from unauthorized access and theft. Furthermore, these attacks can be employed to hijack user sessions and gain unauthorized access to user accounts, highlighting the significance of robust XSS detection in maintaining the integrity of web-based user profiles. By preventing such attacks, web security measures preserve user privacy, as XSS can leak sensitive personal information and shield against potential financial losses that may occur due to financial fraud or unauthorized transactions [3], [6], [10].

Moreover, the importance of XSS detection extends beyond immediate security concerns. XSS attacks can significantly impact an organisation's reputation, leading to a loss of trust and customers. A security breach can tarnish an entity's reputation and have long-lasting consequences. Furthermore, falling victim to XSS attacks may result in legal and regulatory repercussions, particularly in industries where data protection and privacy are subject to strict compliance requirements. Implementing effective XSS detection mechanisms becomes imperative for safeguarding data and adhering to legal and regulatory standards [12]-[14].

Deep learning, as a branch of artificial intelligence, offers a transformative approach to web security, addressing the limitations of traditional XSS detection methods. Conventional methods often struggle to identify evolving and sophisticated attack patterns. Deep learning models, in contrast, exhibit the ability to adapt to these complex patterns and learn from extensive datasets. They can automatically learn relevant features from the data, reducing the need for manual feature engineering. Moreover, deep learning models capture malicious scripts' behaviour, allowing more accurate detection based on script execution patterns [15], [16].

In addition to these advantages, deep learning models offer scalability, making them suitable for handling vast amounts of data generated by web applications. They can be continuously updated and retrained with new data, adapting to emerging threats and ensuring that security measures remain current. Significantly, deep learning can help reduce false positives by distinguishing between benign and malicious behaviors more accurately. It enables real-time detection, improving the responsiveness of web security systems, and can adapt to the evolving landscape of web technologies. In summary, integrating deep learning into web security is crucial for maintaining the integrity and trustworthiness of web applications in an ever-evolving and complex threat landscape [17], [18].

Our primary contributions can be briefly outlined as follows:

- Our paper introduces an adaptable, budget-friendly, and exceptionally efficient cyberattack detection system that harnesses deep learning methods.
- We analyze and evaluate the effectiveness of three deep learning approaches using the XSS Attacks-2019 dataset.

- We provide a systematic and comparative experimental assessment of three deep learning strategies amenable to optimization. This evaluation involves utilizing traditional evaluation metrics, including the confusion matrix, detection accuracy, detection precision, and F1-Score.

## 2. LITERATURE REVIEW

As per reference Ruse and Basu [19], the currently recommended strategy against XSS attacks involves implementing content security policies (CSP). This direct approach aims to heighten security in online communication between users and their devices. CSP primarily focuses on safeguarding secure web services, including crucial information portals, web applications, and internet of things (IoT) networks. It precisely defines communication elements and assets utilized in these services. One notable aspect of CSP is its capacity to swiftly generate clear and compelling reports, promptly notifying administrators about executed attacks.

In a different study [20], a method called "XSSChaser" was introduced, proposing a linear automated approach to prevent XSS attacks within web applications. This technique utilizes chain analysis to identify vulnerable patterns, effectively thwarting XSS attempts. These patterns are crafted through a combination of forward and backward interpretation methods. Meanwhile, Singh [21] presents an intrusion detection system based on containerization. This system employs a query request mapping model to detect and halt XSS attacks. It evaluates impact by analyzing the HTTP load and utilizing the "autobench" tool while assessing performance through metrics such as average page load times, pages per second, memory usage, and processing time.

Another approach [19] demonstrates attacks using tags like "<script>" and "<iframe>" within web requests to target clients' browsers with XSS or SQL queries. Clients access the application via a web server offering a web service, sending web requests through the application's user interface. This triggers database queries and data retrieval for each client. Choi *et al.* [20] investigates two attack types, "N" coding and binary alphabets, proposing a dynamic access control technique to prevent them by leveraging existing detection and prevention technologies.

A distinct framework, Zend [21], addresses XSS attack issues by providing a straightforward security model to protect websites. This model comprises various tools within a sequence of levels for implementation. It integrates the Zend framework web application with the HTML Purifier library, renowned for its capacity to eliminate malicious code responsible for XSS attacks. Additionally, Al-Haija *et al.* [22] introduces an automatic modeling algorithm for the HTML code of e-commerce websites, simplifying HTML code modeling and generating a behavior model stored in an XML file.

Moreover, according to Al-Haija *et al.* [22], emphasis is placed on security vulnerabilities originating from generic entry validation issues leading to XSS attacks. The proposed detection method identifies malicious execution sequences based on a predefined list of legitimate ones and malicious or literal strings generated during a training phase. These lists are organized into four distinct web application execution profiles, each corresponding to a specific attack scenario—the detection module scans for the occurrence of these execution sequences. In recent times, researchers have applied deep learning techniques to the task of detecting XSS attacks. Riera *et al.* [23] introduced an innovative method for representing URL features. They analyzed existing technology for URL attack detection and proposed a deep learning model-based multisource fusion approach. This method enhances the overall accuracy of XSS detection systems and contributes to system stability. Odeh *et al.* [24] also introduced the code-injection detection with deep learning (CODDLE) model designed to combat web-based code injection attacks, including XSS. A vital aspect of this model's innovation involves optimizing the effectiveness of convolutional deep neural networks through a customized preprocessing phase that encodes XSS-related symbols as value pairs. The results demonstrated that this model can significantly enhance the detection rate, achieving recall values of approximately 92%, precision of 99%, and an accuracy rate of 95% compared to a baseline performance.

According to Odeh *et al.* [25], a system design is put forth to detect XSS attacks utilizing an intrusion detection system (IDS). This system uses signatures to recognize these attacks, and a proof-of-concept prototype was developed using the SNORT IDS. The system's implementation depends on rule sets that monitor incoming and outgoing packets, assessing their adherence to predefined rules to identify XSS attacks.

## 3. METHOD

The system put forward in this study was designed using Python version 3.8 and implemented on a high-performance Windows 11 computing system. This computing system comprises a powerful 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz component and 32 GB of RAM. An overview of the development framework is depicted in Figure 2. The system, as envisioned, is divided into four distinct

modules, each with a specific role, before it is deployed for field operation. These modules encompass data collection, engineering, learning, and evaluation. According to the diagram, the model's architecture is structured around the following components:

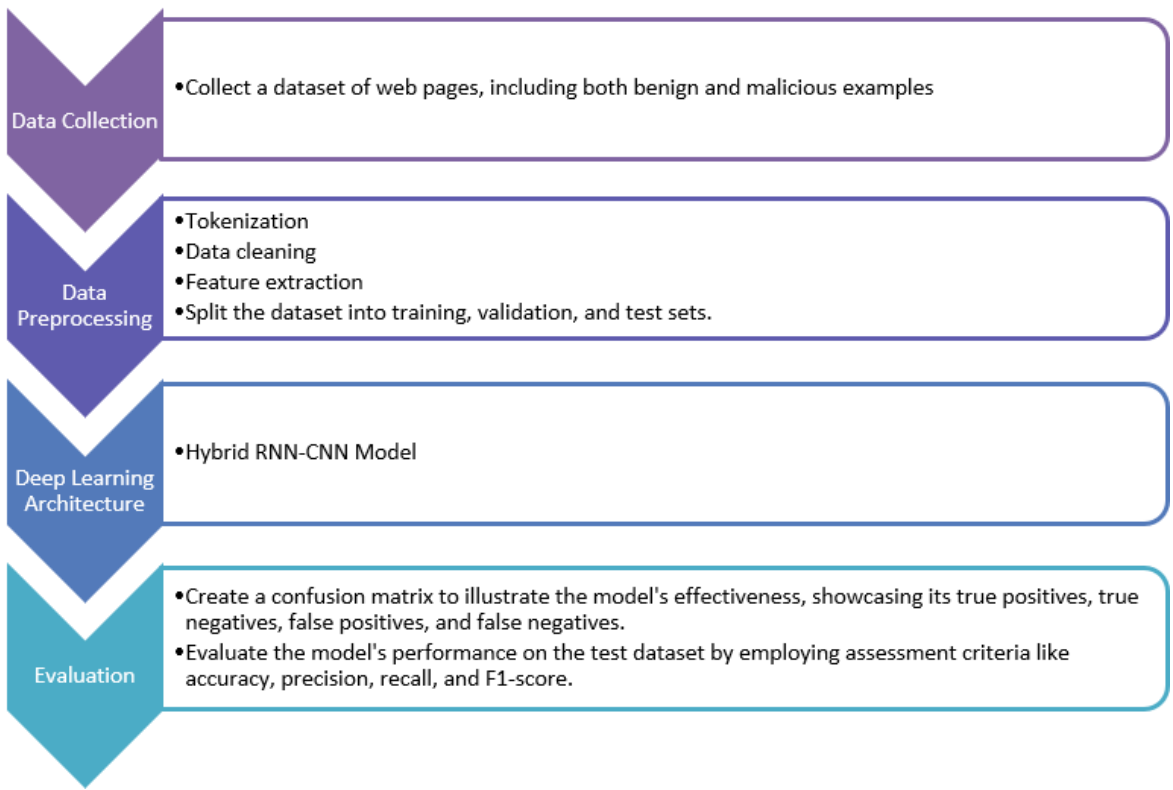


Figure 2. The block diagram for the proposed system

3.1. Data collection

To establish a robust foundation for (XSS) detection, we collected a dataset comprising web pages encompassing benign and malicious instances. The dataset chosen for this endeavor, XSS-2019, includes a balanced distribution of 461 samples, comprising 230 instances classified as anomalies and 231 as benign cases, as shown in Figure 3. These data points are characterized by seven features, which include application names, permissions, application programming interfaces (APIs), website domain name system (DNS) information, internet protocol (I.P.) addresses, geolocation details, and labels denoting their security status.

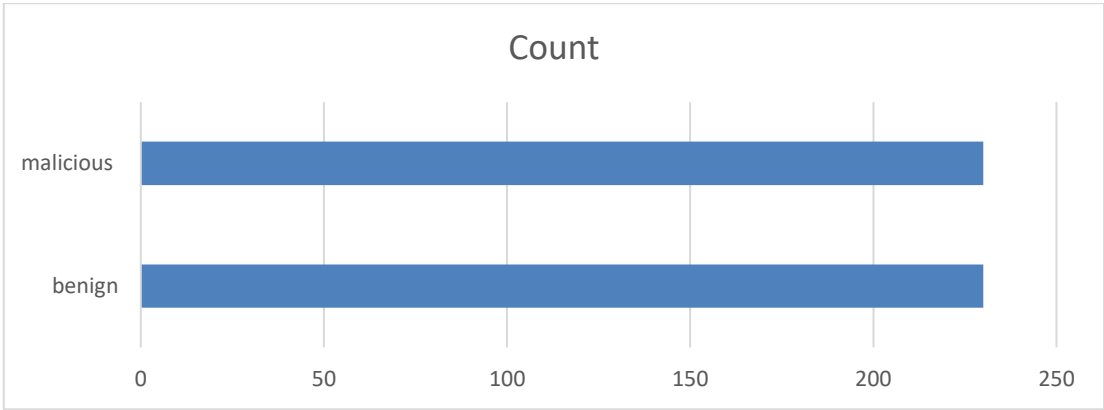


Figure 3. XSS-2019 label balanced distribution

Figure 3 shows a horizontal bar chart that presents the comparative counts of entities classified as either 'malicious' or 'benign'. The chart is structured with the y-axis listing the two categories, with 'malicious' at the top and 'benign' at the bottom. The x-axis represents the count of instances, ranging from 0 to 250. The bar representing 'malicious' extends slightly beyond the 200 mark, indicating a count just over 200, while the 'benign' bar reaches past the halfway point between 150 and 200, suggesting a count that is significantly lower than 'malicious' but exceeds 150. The bars are shaded in a uniform blue color against a clean white background, without any additional embellishments or data labels. The simplicity of the chart's design facilitates immediate comprehension of the disparity between the two classifications. It clearly illustrates that the frequency of 'malicious' instances surpasses that of 'benign' ones within the given dataset. This figure effectively highlights the data's skew towards malicious classifications and may warrant further discussion regarding the implications or causes of this observation within the context of the study.

### 3.2. Data preprocessing

The next crucial step in the dataset preparation involved meticulous preprocessing. This multi-faceted stage encompassed several tasks. First and foremost, tokenization was applied, whereby the text data was effectively split into words or subword tokens, allowing for a granular understanding of the content. Subsequently, data cleaning was carried out, entailing the removal of HTML tags and special characters while simultaneously normalizing the text to ensure uniformity. Finally, feature extraction techniques were employed to convert the textual data into numerical representations for efficient analysis by deep learning models, often leveraging techniques like embeddings.

Figure 4 illustrates a heatmap that represents the correlation coefficients between various features in a dataset. The features included in this heatmap are 'App Names', 'Permissions', 'API Name', 'Website Name', 'IP', 'Location', and 'Label'. Each cell within the heatmap displays the correlation coefficient between the features intersecting at that point, with a scale ranging from -1.0 to 1.0 as indicated by the color bar on the right-hand side of figure.

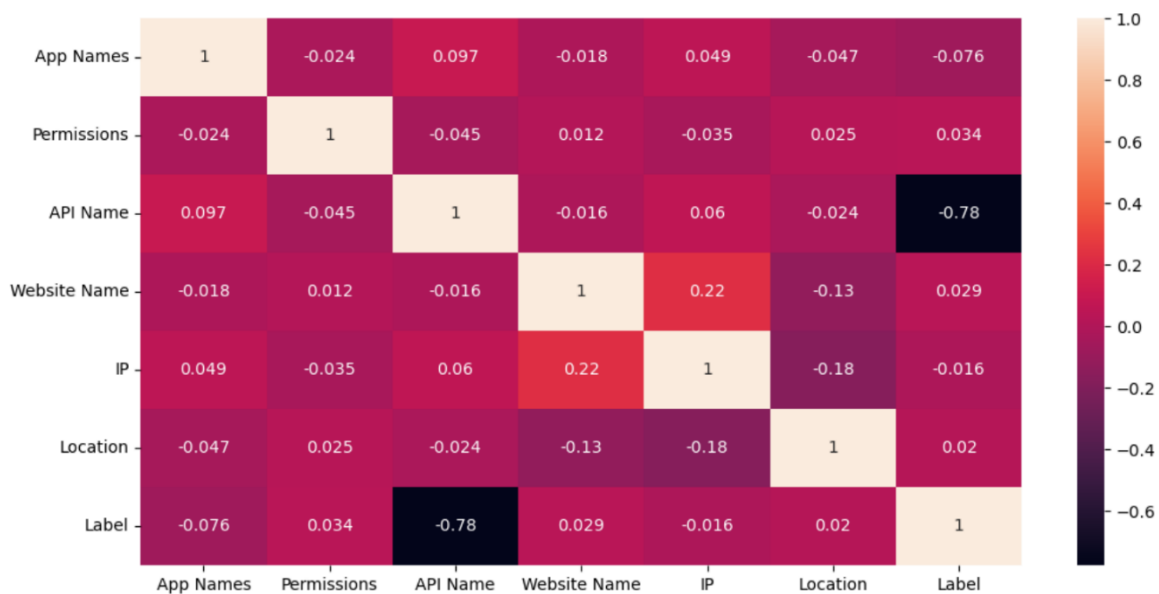


Figure 4. Heatmap using XSS-2019 dataset

The color intensity varies according to the strength and direction of the correlation, with 1 indicating a perfect positive correlation (dark red), 0 indicating no correlation (white), and -1 indicating a perfect negative correlation (dark blue). For instance, the square at the intersection of 'API Name' and 'Label' shows a strong negative correlation with a coefficient of approximately -0.78, marked by a deep blue color. In contrast, most other cells exhibit a very weak correlation, as indicated by the lighter shades of red or white, with coefficients close to zero.

The dataset was thoughtfully divided into three subsets to facilitate model training and evaluation: a training set, a validation set, and a test set. This strategic partitioning ensures that the models can be effectively trained on one portion, validated for performance, and rigorously tested on another. This ensures that the resulting XSS detection system is accurate and robust.

### 3.3. Deep learning architecture-hybrid RNN-CNN model

In pursuit of a more potent (XSS) detection system, we embarked on a novel approach by seamlessly fusing two distinct neural network architectures. The amalgamation combined a recurrent neural network (RNN) with a convolutional neural network (CNN). This tandem of neural networks was meticulously orchestrated to capitalize on their strengths. The RNN, with its inherent capability, was harnessed to apprehend the intricate sequential dependencies intricately woven within HTML and JavaScript code, deciphering the nuanced patterns that might escape other detection mechanisms.

On the other front, CNN, which was celebrated for its proficiency in image analysis, was repurposed to extract features from code snippets and textual content embedded within web pages. By leveraging this blend of RNN and CNN, we could simultaneously capture both the underlying code sequences and the contextual meaning encoded within these web pages. In the culminating phase, the outputs of both networks were harmoniously merged to form the bedrock of our system's final predictions. This innovative amalgamation significantly enhances our ability to discern and flag potential XSS vulnerabilities within web content with precision and depth.

### 3.4. Evaluation

Upon completing the (XSS) detection model's training, a comprehensive evaluation was conducted to gauge its effectiveness in scrutinizing web content for potential threats. This assessment process encompassed multiple vital components. Firstly, a confusion matrix was generated, providing a visually insightful representation of the model's performance, vividly outlining the counts of true positives, true negatives, false positives, and false negatives. This matrix clearly showed how well the model distinguished between benign and malicious instances.

Furthermore, a battery of evaluation metrics was employed to provide a holistic understanding of the model's capabilities. These metrics included accuracy, gauging the overall correctness of the model's predictions; precision, pinpointing the proportion of true positives among the instances predicted as positive; recall, illustrating the model's ability to identify positive instances correctly; and the F1-Score, offering a balanced assessment of precision and recall. These meticulous evaluations culminated in a comprehensive analysis of the model's performance, ensuring its readiness to tackle the challenges of real-world XSS detection scenarios. Figure 5 shows confusion matrix that is commonly used in the evaluation of machine learning algorithms. The matrix compares the actual versus predicted classifications of a binary classifier, with the categories labeled as 'Positive (P)' and 'Negative (N)'.

		<i>Predicted</i>	
		<i>Positive(P)</i>	<i>Negative (N)</i>
<i>Actual</i>	<i>Positive(P)</i>	<i>True Positive (T.P.)</i>	<i>False Negative (F.N.)</i> <i>Type II Error</i>
	<i>Negative (N)</i>	<i>False Positive (F.P.) Type I Error</i>	<i>True Negative (T.N.)</i>

Figure 5. Confusion matrix

The proposed model is assessed using a set of performance metrics derived from the confusion matrix.

$$Recall = \frac{TP}{TP+FN} \quad (1)$$

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

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (3)$$

$$f - measure = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (4)$$

## 4. RESULTS AND DISCUSSION

This section explains the research results and, at the same time, gives. The comprehensive discussion. The Table 1 encapsulates a comprehensive comparative analysis of the performance of various

(XSS) detection models. At the forefront is the "Hybrid RNN-CNN (proposed)" model, representing an innovative approach that combines a (RNN) with a (CNN). This novel fusion resulted in a remarkable accuracy of 96.74%, signifying the model's ability to make correct predictions. Moreover, the "Hybrid RNN-CNN" exhibited an impressive precision score of 97.78%, underscoring its proficiency in correctly identifying positive instances with minimal false positives. While achieving a substantial recall score of 95.65%, the model demonstrated a commendable capacity to capture a significant portion of positive cases. Consequently, its F1-Score of 96.70 attests to its balanced precision and recall.

Table 1. Comparative performance analysis of XSS detection models

Model	Accuracy	Precision	Recall	F1-Score
Hybrid RNN-CNN (proposed)	0.967391	0.977778	0.956522	0.967033
CNN	0.947867	0.970874	0.769231	0.858369
RNN	0.909091	0.909091	0.952381	0.930233

In comparison, while achieving a good accuracy of 94.79%, the standalone CNN model displayed a lower recall score of 76.92%, indicating that it occasionally missed positive instances. Meanwhile, the precision score of 97.08% demonstrated its accuracy in classifying positive instances, but the lower recall affected the F1-Score, which stood at 85.84. On the other hand, the standalone RNN model showcased a decent accuracy of 90.91% and a remarkable recall score of 95.24%. However, the F1-Score of 93.02, although indicating a balance between precision and recall, revealed room for improvement. This comprehensive evaluation highlights the efficacy of the proposed Hybrid RNN-CNN model in XSS detection, offering both high accuracy and a well-balanced trade-off between precision and recall.

Table 2 comprehensively evaluates various (XSS) detection models with key performance metrics, including accuracy, precision, recall, and F1-Score. Notably, our proposed hybrid RNN-CNN model, represented as "proposed," stands out as the frontrunner regarding performance. With an outstanding Accuracy of 96.74%, it showcases unparalleled precision in classifying web content. The precision score of 97.78% underscores its ability to accurately identify positive instances while maintaining minimal false positives. Moreover, the hybrid RNN-CNN model achieves an impressive recall score of 95.65%, indicating its proficiency in capturing a substantial portion of positive instances. This results in an exceptional F1-Score of 96.70, emphasizing the model's balanced trade-off between precision and recall. In direct comparison with the other models in the Table 2, our proposed hybrid RNN-CNN model demonstrates the highest level of effectiveness in identifying potential XSS vulnerabilities within web content.

Table 2. Comparison with other XSS detection models

Reference	Model	Accuracy	Precision	Recall	F1-Score
[26]	Support vector machine (SVM)	0.85130408	0.86044464	0.84173936	0.85098904
[27]	Gated recurrent unit (GRU)	0.84163017	0.85066686	0.83217414	0.84131871
[28]	Naive Bayes (NB)	0.89967363	0.90933354	0.88956546	0.89934069
[29]	Long short-term memory (LSTM)	0.90934754	0.91911132	0.89913068	0.90901102
[30]	SVM+n-Gram	0.88032581	0.88977798	0.87043502	0.88000003
[31]	O-DT	0.8706519	0.8800002	0.8608698	0.8703297
Proposed	Hybrid RNN-CNN	0.967391	0.977778	0.956522	0.967033

## 5. CONCLUSION

Many applications that identify XSS vulnerabilities face several limitations, mainly stemming from the inherent challenges of developing secure applications. Most existing XSS vulnerability scanning tools primarily focus on public areas of internet resources, while vulnerabilities often reside in non-public or less accessible sections. This paper proposes a hybrid model using deep learning algorithms (RNN and CNN). The hybrid RNN-CNN architecture emerges as the frontrunner in our evaluation, exhibiting exceptional performance across all key metrics. With a remarkable accuracy score of 96.74%, it excels in making accurate predictions. The precision score, at an impressive 97.78%, underscores its ability to identify positive instances while minimizing false positives precisely.

Moreover, the model's recall score of 95.65% demonstrates its competence in capturing a significant portion of true positive instances. This, in turn, culminates in a remarkable F1-Score of 96.70, highlighting the model's exceptional balance between precision and recall. In direct comparison with other models in the evaluation, the proposed model unmistakably stands out as a leader, showcasing its superiority in identifying and flagging potential XSS vulnerabilities within web content.



## ACKNOWLEDGEMENTS

The authors sincerely acknowledge the Princess Sumaya University for Technology for supporting steps of this work.

## REFERENCES




- [1] M. S. Vidya and M. C. Patil, "Reviewing effectivity in security approaches towards strengthening internet architecture," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 5, pp. 3862–3871, 2019, doi: 10.11591/ijece.v9i5.pp3862-3871.
- [2] I. Odun-Ayo, W. Toro-Abasi, M. Adebisi, and O. Alagbe, "An implementation of real-time detection of cross-site scripting attacks on cloud-based web applications using deep learning," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 5, pp. 2442–2453, 2021, doi: 10.11591/eei.v10i5.3168.
- [3] M. Alsaffar *et al.*, "Detection of Web Cross-Site Scripting (XSS) Attacks," *Electronics (Switzerland)*, vol. 11, no. 14, 2022, doi: 10.3390/electronics11142212.
- [4] Q. A. Al-Haija, "Cost-effective detection system of cross-site scripting attacks using hybrid learning approach," *Results in Engineering*, vol. 19, 2023, doi: 10.1016/j.rineng.2023.101266.
- [5] F. M. M. Mokbal, W. Dan, W. Xiaoxi, Z. Wenbin, and F. Lihua, "XGBXSS: An Extreme Gradient Boosting Detection Framework for Cross-Site Scripting Attacks Based on Hybrid Feature Selection Approach and Parameters Optimization," *Journal of Information Security and Applications*, vol. 58, 2021, doi: 10.1016/j.jisa.2021.102813.
- [6] J. Kaur, U. Garg, and G. Bathla, "Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 12725–12769, 2023, doi: 10.1007/s10462-023-10433-3.
- [7] M. M. Hassan, B. R. Ahmad, A. Esha, R. Risha, and M. S. Hasan, "Important factors to remember when constructing a cross-site scripting prevention mechanism," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 2, pp. 965–973, 2022, doi: 10.11591/eei.v11i2.3557.
- [8] S. Kascheev and T. Olenchikova, "The Detecting Cross-Site Scripting (XSS) Using Machine Learning Methods," *Proceedings - 2020 Global Smart Industry Conference, GloSIC 2020*, pp. 265–270, 2020, doi: 10.1109/GloSIC50886.2020.9267866.
- [9] R. W. Kadhim and M. T. Gaata, "A hybrid of CNN and LSTM methods for securing web application against cross-site scripting attack," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 2, pp. 1022–1029, 2020, doi: 10.11591/ijeecs.v21i2.pp1022-1029.
- [10] M. Singh, P. Singh, and P. Kumar, "An Analytical Study on Cross-Site Scripting," *2020 International Conference on Computer Science, Engineering and Applications, ICCSEA 2020*, 2020, doi: 10.1109/ICCSEA49143.2020.9132894.
- [11] H. Pan, Y. Fang, C. Huang, W. Guo, and X. Wan, "GCNXSS: An Attack Detection Approach for Cross-Site Scripting Based on Graph Convolutional Networks," *KSI Transactions on Internet and Information Systems*, vol. 16, no. 12, pp. 4008–4023, 2022, doi: 10.3837/tiis.2022.12.013.
- [12] W. Melicher, A. Das, M. Sharif, L. Bauer, and L. Jia, "Riding out DOMsday: Toward Detecting and Preventing DOM Cross-Site Scripting," *25th Annual Network and Distributed System Security Symposium, NDSS 2018*, 2018, doi: 10.14722/ndss.2018.23309.
- [13] I. S. Joshi and H. J. Kiratsata, "Cross-Site Scripting Recognition Using LSTM Model," *International Conference on Intelligent Computing and Communication*, pp. 1–10, 2023, doi: 10.1007/978-981-99-1588-0\_1.
- [14] B. Peng, X. Xiao, and J. Wang, "Cross-Site Scripting Attack Detection Method Based on Transformer," *2022 IEEE 8th International Conference on Computer and Communications, ICC3 2022*, pp. 1651–1655, 2022, doi: 10.1109/ICC356324.2022.10065892.
- [15] J. M. Gan, H. Y. Ling, and Y. B. Leau, "A Review on Detection of Cross-Site Scripting Attacks (XSS) in Web Security," *Communications in Computer and Information Science*, vol. 1347, pp. 685–709, 2021, doi: 10.1007/978-981-33-6835-4\_45.
- [16] B. Buz, B. Gülççek, and Ş. Bahtiyar, "A Hybrid Machine Learning Model to Detect Reflected XSS Attack," *Balkan Journal of Electrical and Computer Engineering*, vol. 9, no. 3, pp. 235–241, 2021, doi: 10.17694/bajece.927417.
- [17] T. A. Taha and M. Karabatak, "A proposed approach for preventing cross-site scripting," *6th International Symposium on Digital Forensic and Security, ISDFS 2018 - Proceeding*, vol. 2018-January, pp. 1–4, 2018, doi: 10.1109/ISDFS.2018.8355356.
- [18] K. Pranathi, S. Kranthi, A. Srisaila, and P. Madhavilatha, "Attacks on Web Application Caused by Cross Site Scripting," *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018*, pp. 1754–1759, 2018, doi: 10.1109/ICECA.2018.8474765.
- [19] M. E. Ruse and S. Basu, "Detecting cross-site scripting vulnerability using concolic testing," *Proceedings of the 2013 10th International Conference on Information Technology: New Generations, ITNG 2013*, pp. 633–638, 2013, doi: 10.1109/ITNG.2013.97.
- [20] J. H. Choi, C. Choi, B. K. Ko, and P. K. Kim, "Detection of cross site scripting attack in wireless networks using n-Gram and SVM," *Mobile Information Systems*, vol. 8, no. 3, pp. 275–286, 2012, doi: 10.3233/MIS-2012-0143.
- [21] T. Singh, "Detecting and Prevention Cross-Site Scripting Techniques," *IOSR Journal of Engineering*, vol. 2, pp. 854–857, 2012.
- [22] Q. A. Al-Haija, A. Odeh, and H. Qattous, "PDF Malware Detection Based on Optimizable Decision Trees," *Electronics (Switzerland)*, vol. 11, no. 19, 2022, doi: 10.3390/electronics11193142.
- [23] T. S. Riera, J. R. B. Higuera, J. B. Higuera, J. J. M. Herraiz, and J. A. S. Montalvo, "A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques," *Computers and Security*, vol. 120, 2022, doi: 10.1016/j.cose.2022.102788.
- [24] A. Odeh, I. Keshta, and E. Abdelfattah, "Machine Learning Techniques for Detection of Website Phishing: A Review for Promises and Challenges," *2021 IEEE 11th Annual Computing and Communication Workshop and Conference, CCWC 2021*, pp. 813–818, 2021, doi: 10.1109/CCWC51732.2021.9375997.
- [25] A. Odeh, I. Keshta, and E. Abdelfattah, "Efficient detection of phishing websites using multilayer perceptron," *International Journal of Interactive Mobile Technologies*, vol. 14, no. 11, pp. 22–31, 2020, doi: 10.3991/ijim.v14i11.13903.
- [26] G. E. Rodriguez, J. G. Torres, P. Flores, and D. E. Benavides, "Cross-site scripting (XSS) attacks and mitigation: A survey," *Computer Networks*, vol. 166, 2020, doi: 10.1016/j.comnet.2019.106960.
- [27] P. Sriramya, S. Kalaiarasi, and N. Bharathi, "Anomaly Based Detection of Cross Site Scripting Attack in Web Applications Using Gradient Boosting Classifier," *Communications in Computer and Information Science*, vol. 1394 CCIS, pp. 243–252, 2021, doi: 10.1007/978-981-16-3653-0\_20.






- [28] Y. Lin, O. T. Onadele, and X. Gu, "CDL: Classified Distributed Learning for Detecting Security Attacks in Containerized Applications," *ACM International Conference Proceeding Series*, pp. 179–188, 2020, doi: 10.1145/3427228.3427236.
- [29] A. Niakanlahiji and J. H. Jafarian, "WebMTD: Defeating Cross-Site Scripting Attacks Using Moving Target Defense," *Security and Communication Networks*, vol. 2019, 2019, doi: 10.1155/2019/2156906.
- [30] P. I. Radoglou-Grammatikis and P. G. Sarigiannidis, "Securing the Smart Grid: A Comprehensive Compilation of Intrusion Detection and Prevention Systems," *IEEE Access*, vol. 7, pp. 46595–46620, 2019, doi: 10.1109/ACCESS.2019.2909807.
- [31] A. W. Marashdih, Z. F. Zaaba, and K. Suwais, "Predicting input validation vulnerabilities based on minimal SSA features and machine learning," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 10, pp. 9311–9331, 2022, doi: 10.1016/j.jksuci.2022.09.010.

## BIOGRAPHIES OF AUTHORS



**Ammar Odeh**    received his Ph.D. from University of Bridgeport (UB), USA, in 2015. He is an Associate Professor at the Department of Computer Science, Faculty of King Hussein School of Computing Sciences, Princess Sumaya University for Technology, Amman, Jordan. His research interests include cybersecurity, cryptography, and the internet of things (IoT). He can be contacted at email: a.odeh@psut.edu.jo.



**Anas Abu Taleb**    is an associate professor in the Department of Computer Science at Princess Sumaya University for Technology, Amman, Jordan. He received a Ph.D. in Computer Science from the University of Bristol, UK 2010, an M.Sc. in Computer Science from the University of the West of England, UK, 2007 and a B.Sc. Degree in Computer Science from Princess Sumaya University for Technology, Jordan, 2004. He has published several journal and conference papers on sensor networks. In addition to sensor networks, he is interested in network fault tolerance, routing algorithms, and mobility models. He can be contacted at email: a.abutaleb@psut.edu.jo.