Available online at sjuoz.uoz.edu.krd

journals.uoz.edu.krd

Science Journal of University of Zakho

Vol. 13, No.1 pp58–64January-March,2025

p-ISSN: 2663-628X
e-ISSN: 2663-6298

WEB VULNERABILITIES DETECTION USING A HYBRID MODEL OF CNN, GRU AND ATTENTION MECHANISM

Sarbast H. Ali [1], Arman I. Mohammed [1], Sarwar MA. Mustafa [2], Sardar Omar Salih [3, *]

[1.] Duhok Technical College, Duhok Polytechnic University.

[2.] College of Science, University of Duhok.

[3.] Duhok Technical Institute, Duhok Polytechnic University.

Corresponding author email: sardar@dpu.edu.krg

Received: 14 Oct 2024 / Accepted:27 Nov., 2024/ Published: 12 Jan., 2025. https://doi.org/10.25271/sjuoz.2024.12.3.1404

ABSTRACT

The frequency of cyber-attacks has been rising in recent years due to the fact that startup developers have failed to overlook security issues in the core web services. This stated serious concerns about the security of the web. Therefore, this paper proposes a hybrid model built on the base of Convolutional Neural Networks (CNN), Gated Recurrent Units (GRU) and an attention mechanism to detect vulnerabilities in application code. Particularly, the model can help detect attacks based on Structured Query Language Injection (SQLi), Cross-Site Scripting (XSS), and command injection. When using the dataset SXCM1, our model achieved 99.77%, 99.66% and 99.63% for training, validation and testing, respectively. The results obtained on data from the DPU-WVD dataset are even better because it was 99.97%, 99.98% and 99.99% for training, validation and testing, respectively. These results significantly outperform the state-of-the-art models and can strongly identify vulnerabilities in web applications. Through training, on both the SXCM1 and DPU-WVD datasets, the model achieved an accuracy rate of 99.99%. The results show that this combination model is highly effective at recognizing three vulnerability categories and surpasses cutting-edge models that usually specialize in just one type of vulnerability detection.

KEYWORDS: CNN, Web vulnerabilities, deep learning, XSS, SQL injection.1.

1. INTRODUCTION

Code injection attacks let attackers run malicious code inside a program, therefore compromising the security and integrity of web applications. The reliability and trustworthiness of software products depend on detecting and reducing such weaknesses. This research aims to build a strong solution for automated code injection detection using deep learning methods, thus enhancing the security posture of software applications. The front-end applications and backend databases are susceptible to numerous attacks due to their internet accessibility. Especially noteworthy and often ranking among the top 10 vulnerabilities found by OWASP (OWASP Top Ten | OWASP Foundation, 2021). The frequency of these attacks has grown by more than 300% in recent years as attackers progressively use sophisticated methods such as encryption and obfuscated code to avoid discovery (Alarfaj *et al*., 2023).

SQLi is a web security issue that allows an attacker to interfere with the queries an application submits to its database. Typically, it lets attackers read information they would not usually be able to obtain. This could include user-owned data and any other data the application can access. An assailant can regularly change or eliminate this data, therefore affecting the content or functionality of the program in a constant manner. An attacker, in particular circumstances, may utilize an escalated SQLi attack or a denial-of-service attack to compromise the underlying server or other back-end infrastructure. In-band SQLi is the most often occurring and basic SQLi attack. As can be seen in Figure 1, in-band SQLi is the condition whereby an attacker can start an attack and gather data utilizing the same communication channel. The two most common variants of in-band SQLi are error-based and union-based SQLi. (SQL Injection | OWASP Foundation, 2023).

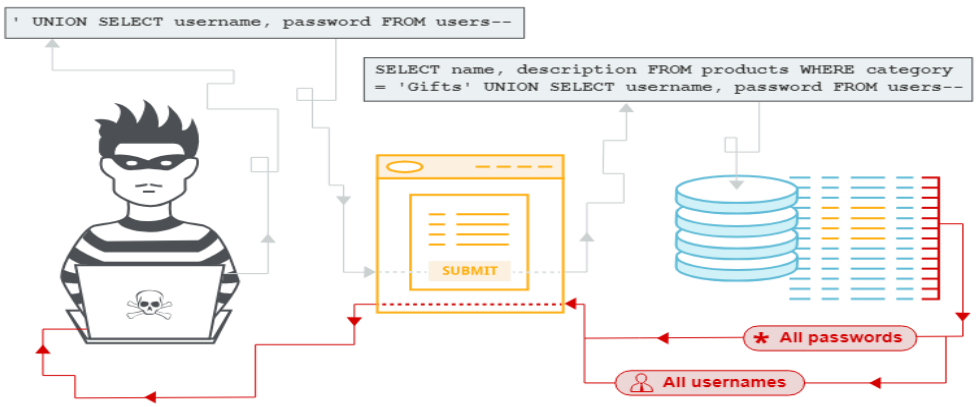

Figure 1: In-band SQLi Attack Diagram (Payloadbox/Sql-Injection-Payload-List: SQL Injection Payload List, 2021)

* Corresponding author

Union-based SQLi combines the output of two or more SELECT queries into a single result that is returned as part of the HTTP response. Error-based SQLi is an in-band SQLi method that uses the database server's error messages to gather details about the database's structure. In some cases, an attacker can use error-based SQLi to enumerate the entire database structure and access all the data (Blind SQL Injection | OWASP Foundation, 2023).

Additionally, there are many other types of SQLi that make high potential risks to web application security, such as (Inferential SQLi, Boolean-based, Time-based Blind SQLi, Out-of-band SQLi and Voice Based SQLi). XSS is a high-risk online vulnerability that arises when malevolent scripts are injected into websites that are otherwise reliable and secure. XSS attacks occur when a hacker transmits malicious code, usually in the form of a browser-side script, to another user via an online application. These vulnerabilities are prevalent and occur when a web application neglects to adequately authenticate or encode user input before including it in its output. An unidentified user may be subjected to a harmful script sent by an assailant employing XSS. The script will execute because the end user's browser lacks the capability to ascertain the level of trustworthiness of the script. This malicious script has the ability to get any cookies, session tokens, or other confidential information that is kept by the browser and associated with the website. This is possible because the script is perceived as trustworthy, originating from a source named KirstenS in the year 2022. The XSS vulnerability can be classified into two types: server XSS and client XSS. The level of risk associated with both types depends on the specific scripts that have been implemented XSS | OWASP Foundation, 2022). Furthermore, command injection attacks seek to exploit vulnerabilities in an application in order to run unauthorized instructions on the underlying operating system. Command injection attacks can occur when an application transmits a system shell along with sensitive user information, such as forms, cookies, and HTTP headers. In this attack, the attacker often executes operating system commands using the privileges of the vulnerable software. Inadequate input validation is a significant contributing element to the potential occurrence of command injection attacks. This attack distinguishes itself from code injection by enabling the attacker to embed personalized code that the program will eventually run. Command injection enables an attacker to augment the inherent capabilities of a program by executing system commands without manually inserting code (Command Injection | OWASP Foundation, 2021).

This paper aims to design robust deep learning methods that can accurately predict and categorise different code flaws (SQLi, XSS, Command Injection, benign code segments). This is possible by varied input data consisting of coding snippets and achieving strong predictions because of using codified features as inputs that can be dynamic text, a static vector of tokens, or both, and using a collection of convolutional neural network layers and recurrent neural network layers.

This paper is organized as follows: Section 2 reviews related works, providing a foundation for the study. Section 3 details the data collection and preprocessing methods employed. Section 4 introduces the proposed approach, outlining the novel techniques implemented. Section 5 presents the results, analyzing the effectiveness of our approach. Finally, Section 6 concludes the paper, summarizing the key findings and suggesting directions for future research.

2. THEORETICAL BACKGROUND

(Arasteh et al., 2024) Proposed a robust technique for identifying SQLi attacks in online applications, to improve the reliability, exactness, and responsiveness of the detection process. (Kakisim, 2024) developed a new technique called "Bidirectional LSTM-CNN based on Multi-View Consensus" (MVC-BiCNN) that uses deep learning to detect SQLi threats. (Tadhani et al., 2024) introduced an innovative method to improve the security of online applications against SQLi and XSS assaults by implementing deep learning techniques. The new hybrid deep learning model integrates Convolutional Neural Networks (CNNs) to extract features and Long Short-Term Memory (LSTM) networks to capture connected relationships in sequential data. (Younas *et al.,* 2024) proposed a highly effective artificial intelligence method for promptly identifying XSS attacks in web applications using Long Short-Term Memory (LSTM). (Sethi *et al.,* 2023) presented a flexible and scalable method that uses deep learning techniques to promptly detect and mitigate XSS attacks in web applications. The research shows the advantages of the Multilayer Perceptron (MLP) architecture by analyzing different deep learning models. (Abhishek *et al.,* 2023) introduced a mixed architecture that combines CNN and Machine Learning (ML) to identify instances of XSS. The framework is designed to address the pervasive problem of XSS attacks in web applications by providing a strong and accurate method for identifying and categorizing. The study conducted by (Nilavarasan *et al.,* 2023) focuses on applying CNN to detect XSS vulnerabilities in web applications when they achieved significant levels of accuracy, precision, and recall in their findings. (Tan *et al*., 2023) proposed the PATS model, which utilizes abstract syntax trees and attention mechanisms to identify XSS vulnerabilities, and this approach led to high accuracy and enabled proactive security. The study work by (Mondal *et al.,* 2023) tried to prevent adversarial attacks by focusing on enhancing the detection of XSS by utilizing one of the reinforcement learning approaches known as Trust Region Policy Optimization (TRPO). Another fascinating work focused on identifying SQLi threats using Natural Language Processing (NLP) and Deep Learning (DL) techniques, which was done by (Natarajan *et al.,* 2023). They used CNN to indicate a greater performance than classical Machine Learning (ML) algorithms (Sun *et al.,* 2023). Enhanced the detection performance by implementing the combination of Text CNN and Bi-LSTM neural networks, and further improvement was achieved by incorporating an attention mechanism and word embeddings via pre-trained BERT vectors. (Yan *et al.,* 2022) presented an advanced MRBN-CNN model modified for detecting XSS attacks by displaying a high standard of accuracy metrics. A deep-learning approach for identifying web injection strikes with the mean of merging features extracted from HTTP and URL request bodies to construct a multi-classification model was introduced by (Zhao *et al.,* 2022). Mondal *et al.* (2022) recommended using reinforcement learning to bolster XSS detection and counter adversarial attacks by fortifying the algorithm's protection to assaults. (Abdulhamz *et al.,* 2022) introduced a 2D-CNN model customized specifically for SQLi detection on a designated dataset. Deploying deep learning algorithms, this model autonomously extracts features from SQL queries, converting them into a two-dimensional matrix for classification purposes. (Roy *et al.,* 2022) focused on identifying SQLi attacks using machine learning classifiers like Logistic Regression, AdaBoost, Random Forest, Naive Bayes, and XGBoost. (Ashlam *et al.,* 2022) presented a multi-phase algorithmic framework that integrates advanced machine learning and deep learning techniques to mitigate SQLi attacks and elevate real-time database security. (Zhang *et al.,* 2022) presented a specialized deep neural network model to identify SQLi attacks. This achievement can be due to various reasons, such as transforming data into word vectors, employing ReLU functions, optimizing loss functions, and including Dropout to improve generalization abilities. (Demilie *et al.,* 2022) conducted a study on SQLi attacks in web applications. They developed a comprehensive framework to detect and prevent these attacks using a combination of machine learning (ML) algorithms and classical methods such as Naive Bayes, Decision Trees, Support Vector Machines, Random Forests, Logistic Regression, and Multilayer Perceptron-based Neural Networks. (Niu *et al*, 2020) propose a hybrid CNN-GRU model aimed at enhancing the

accuracy of web attack detection. By capturing spatial and sequential patterns in network data, their approach improves detection performance and processing speed compared to traditional methods. Similarly, (Jiang *et al.,* 2021) introduce a CNN-GRU-Attention model specifically for detecting malicious domains. Adding an attention mechanism helps the model focus on critical features within domain names, yielding higher detection precision and increased robustness against evolving threats. Together, these studies highlight the potential of CNN-GRU architectures, particularly when combined with attention mechanisms, to enhance security applications.

Data Collection

Two distinct datasets are employed in the current study. The primary dataset is designated as the SQL Injection XSS Command Injection Mix Dataset. Version 1.0.0 (SXCM1), introduced by the SQLi XSS Dataset in 2023 (SQLi XSS Dataset, 2023), comprises 206,636 distinct code fragments. We also create a hybrid dataset (DPU-WVD) comprising about 1,003,996 code words written by humans and machines. AI produces one million lines of code, whereas 3,996 lines are sourced from various payloads on GitHub. The pseudocode utilized to produce the AI-generated code is as follows.

Pseudo Code:

```
START
DEFINE FUNCTION generate_random_string(length)
        RETURN random string of letters and digits of given length
DEFINE FUNCTION generate_payload(index, payloads)
        RETURN payloads[index % length of payloads]
DEFINE FUNCTION generate_normal_input()
        IF random number < 0.3 THEN RETURN random benign SQL query
        ELSE IF random number < 0.6 THEN RETURN random benign XSS string
        ELSE RETURN random benign command input
SET num_samples to 250000
OPEN 'web_vulnerabilities_dataset.csv' FOR writing
        WRITE header ['Sentence', 'SQLInjection', 'XSS', CommandInjection', 'Normal']
        FOR i FROM 0 TO num_samples - 1
        WRITE row with generate_payload(i, SQLi payloads) as Sentence, 1, 0, 0, 0
        WRITE row with generate_payload(i, XSS payloads) as Sentence, 0, 1, 0, 0
        WRITE row with generate_payload(i, command injection payloads) as Sentence, 0, 0, 1, 0
        WRITE row with generate_normal_input() as Sentence, 0, 0, 0, 1
CLOSE file
END
```

The DPU-WVD includes highly challenging examples from which the network can learn. For instance, the command ls -l n27z8 is marked as vulnerable to command injection, while ls -l R8wnZ is marked as normal. This suggests that in the dataset, the filename "n27z8" might be interpreted by the system in such a way that it allows additional commands to be injected and executed, making it vulnerable to command injection attacks. In contrast, the filename "R8wnZ" does not display this behavior and is regarded as secure or typical. Data preparation covers multiple sequential steps. The main purpose of the proposed method was to separate characters from symbols. The procedure involved eliminating unwanted characters, addressing disparities in the dataset, making important user data such as username, password and proprietary code, and eliminating comments.

Concept And Methods

The proposed methodology involves preprocessing the training data by transforming each character of the words in the dataset into vectors, as well as constructing a hybrid model of CNN, GRU, and the multi-head attention mechanism.

Vectorizing Characters

This research presents a highly effective method for identifying web vulnerabilities, such as SQLi, XSS, command injection, and normal code segments. We aim to develop a highly efficient AI model to categorise the input code accurately. The incoming text performs preprocessing using several functions. The initial phase entails dividing the input text into individual letters and symbols utilizing two different procedures. Throughout this step, cleaning and eliminating unnecessary sentences, such as comments, is implemented.

The initial function exclusively retrieves characters from the input text. It changes a list of strings X (input data) into sequences of character indices using a predefined alphabet. Subsequently, it adjusts these sequences to a fixed length by padding or truncating them to a maximum length of *max_len*. Initially, the function establishes a set of legal characters known as an alphabet. "mat" is a blank list determined to record the indications of valid characters for each string in *X* variable. If the is_remove_comment flag is set to True, the comments within the string are removed. Then, every character is transformed into lowercase, and if it belongs to the alphabet, its position is added to the mat array. Upon completion of processing, the variable "mat" is added to the results list. This technique is repeated to extract symbol tags from the provided input text.

Hybrid Model of CNN And GRU

The paper introduces a new model for detecting web vulnerabilities. Our proposed model is a multilevel architecture developed to efficiently manage and interpret complicated input data. Furthermore, it utilizes a fusion of embedding layers, CNNs, bidirectional GRUs, and multi-channel attention mechanisms to handle and gain knowledge from textual and symbolic input. The architecture combines CNNs to extract inner features and GRUs to model sequential dependencies. Additionally, the attention mechanism is utilized to rank relevant segments of the sequences, as illustrated in Figure 2. The primary objective of this comprehensive technique is to achieve a powerful capability in identifying web vulnerabilities by analyzing a wide range of complicated input data.

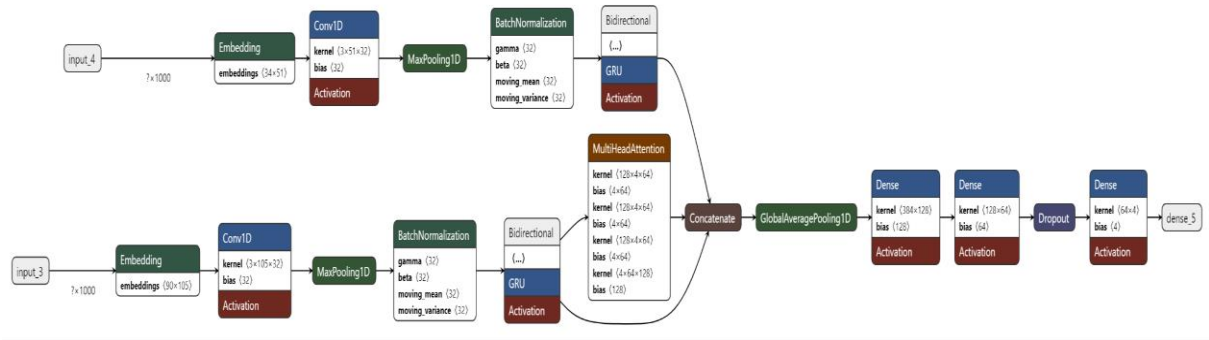

Figure 2: Proposed Hybrid Model of CNN and GRUs.

The model launches by analyzing two separate categories of inputs. The primary input is in the form of text, which has the potential to include various types of textual information, such as code snippets, error messages, or descriptions that may include vulnerabilities. The second input consists of symbols that act as representations of specific tokens or symbols that are significant in the context of web vulnerabilities. These symbols may include special characters found in code or configuration files. Embedding layers, as described by (Mikolov *et al.*, 2013) transform inputs that are high-dimensional and sparse, such as integers, into vectors that are dense and few-dimensional. The function of this layer is crucial in decreasing the number of dimensions by converting inputs into a more easily manageable format that the neural network can process efficiently. Moreover, feature learning is crucial since it enables the model to acquire representations of the input data in which related inputs possess comparable representations, improving pattern recognition.

CNNs have multiple functions within the model. Firstly, they demonstrate exceptional proficiency in extracting features and identifying local patterns and characteristics within the input sequences, such as certain sequences of tokens frequently observed in vulnerability signatures. In addition, CNNs assist in reducing the dimensionality of data by incorporating max pooling layers. These layers effectively decrease the complexity of the input, resulting in more efficient computations in succeeding layers while preserving the most significant properties. Moreover, regularization techniques such as batch normalization and dropout layers are utilized to mitigate the problems of overfitting and underfitting. Batch normalization standardizes the inputs to subsequent layers, whereas dropout layers stochastically eliminate some neurons during training, improving the model's resilience and capacity to generalize. The hybrid model utilizes a recurrent neural network layer, notably GRUs (Cho *et al.*, 2014), to effectively capture the sequential relationships present in the data. GRUs that update and reset gates allow the model to analyze input sequences bidirectionally, effectively capturing dependencies that may occur in both forward and backward directions. Bidirectional processing is essential for comprehending context in sequences. GRUs are particularly adept at understanding the time-based patterns and extended connections in sequences commonly found in web application logs, user actions, and code execution pathways. Including a multi-head attention mechanism (Vaswani *et al.*, 2017) improves the model's capabilities in various aspects. Firstly, attention enables the model to assign varying levels of importance to distinct sections of the input sequences, prioritizing the most pertinent portions that are probably to highlight vulnerabilities. In addition, the model can enhance its knowledge of the data by using numerous heads to capture different types of interactions and correlations between different regions of the sequences, as shown in Figure 3.

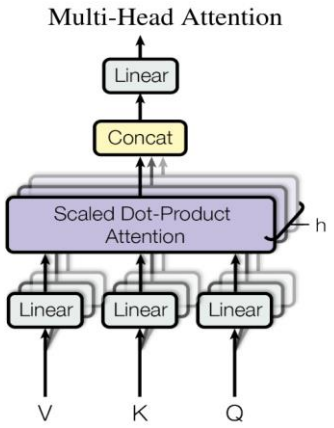

Figure 3: Multi-Head Attention (Vaswani *et al.*, 2017).

The concatenation and global average pooling layers in combining layers have two primary functions. To begin with, they consolidate information from various viewpoints by merging the results of the GRU layers with the attention mechanism. Additionally, global average pooling reduces the sequence data into a vector of a constant size, effectively decreasing the number of dimensions and making the input more feasible for the dense layers.

The utilization of these layers enhances the efficiency of the model by integrating CNNs, GRUs, and attention mechanisms, resulting in multiple benefits. Firstly, it does a thorough analysis by capturing both local and global patterns in the data. Moreover, the model's flexibility is improved by incorporating trainable embedding layers and attention mechanisms, which allow it to easily adapt to various types of input data and successfully learn important features. In addition, the model's robustness is enhanced by regularization techniques such as dropout and batch normalization, which reduce overfitting and enhance performance on unfamiliar data. Moreover, the utilization of bidirectional GRUs and attention mechanisms guarantees a comprehensive understanding of the context, allowing the model to precisely detect vulnerabilities by comprehending the context of input sequences.

3. RESULTS AND DISCUSSION

In this section, we provide and examine the results of this study, establishing connections between them and the previously published hypothesis. The results are methodically analyzed to identify patterns, anomalies, and important trends. After presenting the data using figures and tables, we proceed to have a thorough discussion in which we place our findings within the context of existing literature and theoretical frameworks.

Training Process

The proposed methodology was trained using both datasets (SXCM1 and DPU-WVD) on the online deep-learning platform Kaggle, utilizing two T4 GPUs. The training process of the first dataset took about 96 minutes and 66 seconds to finish 100

epochs while the second dataset took about 115 minutes to finish the training process of 20 epochs. Both datasets are trained with the hyperparameters defined in Table 1. The datasets were split into 80% for the training set and 20% for the testing set. The training set was split into 80% for training and 20% for validation.

Table 1: Hyperparameter Initialization

Hyperparameter	**Initialization**
Input Size (*Max_len*)	1 * 1000
Optimizer	Adam
Learning Rate	0.001
Loss Function	Categorical Cross-entropy
Batch Size	64
Epochs	20-100
Dropout	0.3 (throughout network)

Training Results

The proposed model showed impressive results in terms of accuracy and loss for both datasets, as illustrated in Figures 4 and 5. The SXCM1 dataset gained a training accuracy of 96.57% and a validation accuracy of 94.35% in the first epoch, which is impressive. The DPU-WVD dataset, on the other hand, stated higher starting values, with training and validation accuracies reaching 99.75% and 95.84%, respectively.

During the training process, the model's performance exhibited continuous improvement in accuracy and loss for both datasets. In the SXCM1 dataset, the training loss started at a low value of 0.1061, which coincides with the high initial training accuracy of 96.57%. This suggests that the model rapidly comprehended the patterns in the data. Nevertheless, the initial validation accuracy of 94.35% indicated the presence of overfitting. However, this issue was addressed and reduced as the training advanced by implementing dropout and batch normalization techniques in the proposed model. As a result, the training and validation accuracies peaked at 99.78% and 99.67% respectively, indicating the model's strong ability to learn and generalize.

In the case of the DPU-WVD dataset, the model initially exhibited much greater accuracy for training and validation. The training procedure enabled a seamless and consistent improvement in performance indicators. The model effectively adjusted to the complex elements of the dataset, ultimately obtaining training and validation accuracies of 99.97% and 99.98% accordingly by the conclusion of epoch 20. The significant enhancement highlights the model's ability to acquire knowledge from a wide range of data distributions and well manage different levels of starting correctness. Despite the DPU-WVD dataset being larger than SXCM1, the hybrid model demonstrated faster learning capabilities. This can be attributed to the clarity and uniformity of the code phrases created using AI. Figures 4 and 5 visually represent this progressive enhancement in performance. They clearly illustrate the upward trajectory of the training and validation accuracies over successive epochs for both datasets. The consistent increase in accuracy metrics highlights the effectiveness of the proposed model's architecture and training regimen in achieving optimal results across different datasets.

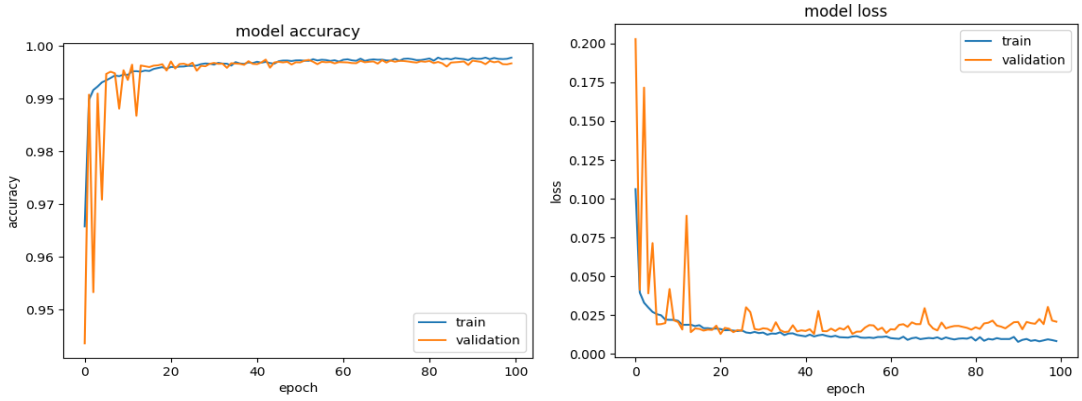

Figure 4: Training and loss metrics for SXCM1 dataset.

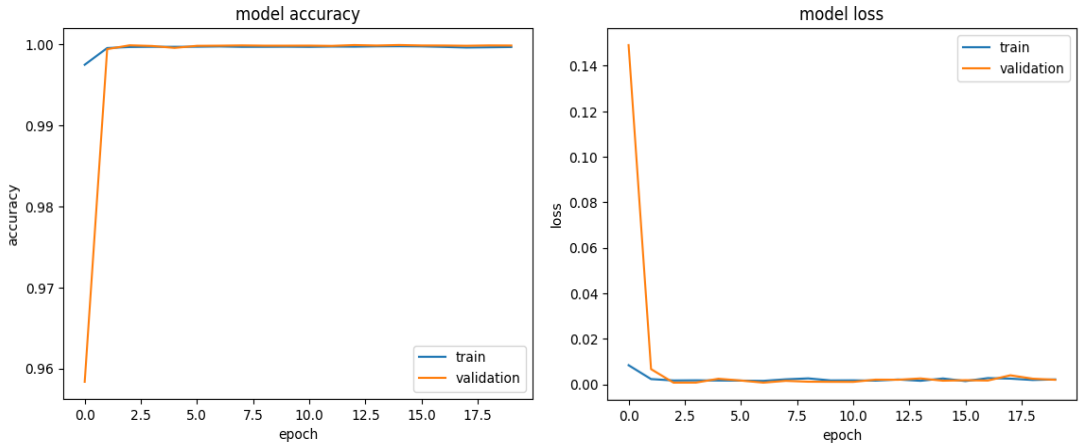

Figure 5: Training and loss metrics for DPU-WVD dataset.

The model performs well on both datasets, achieving high accuracy and low loss. For the SXCM1 dataset, there's some fluctuation in validation accuracy and loss, as seen in Figure 4, suggesting slight overfitting and room for improvement. Techniques like regularization or data augmentation could help stabilize the results. In contrast, the DPU-WVD dataset shows

smooth and consistent performance, with both training and validation accuracy reaching near-perfect levels quickly, as illustrated in Figure 5, indicating strong generalization. Overall, the model handles DPU-WVD data very well, while a few adjustments might improve stability on SXCM1.

Testing Results

The testing phase is crucial for assessing a model's efficiency and robustness when evaluated with unseen data. This phase serves as a key evaluation metric to demonstrate how the proposed model will perform in production. Table 2 provides information on both datasets in terms of accuracy, precision, and recall. Notably, the hybrid model achieved an accuracy of 99.63% for the SXCM1 dataset and 99.99% for our proposed dataset

Table 2: Testing metrics for both dataset

Metric	SXCM1 dataset	DPU-WVD dataset
Accuracy	99.63%	99.99%
Precision	99.66%	99.98%
Recall	99.63%	99.99%

The suggested hybrid model has been evaluated against contemporary cutting-edge models for various tasks, such as XSS, SQLi, and Command Injection detection. Our hybrid model is capable of detecting all three sorts of vulnerabilities, unlike earlier models that only detect a single type. In addition, it has the capability to detect regular language that does not have any weaknesses, resulting in improved precision. The exceptional result is exemplified in Table 3, showcasing the efficacy and adaptability of our methodology.

Table 3: Comparison of the proposed model versus benchmark models

Model		XSS	SQL injection	Command Injection	All
(Arasteh *et al.*, 2024)			99.68%		
(Kakisim, 2024)			99.96%		
(Younas *et al*., 2024)		99.00%			
(Abhishek *et al*., 2023)		99.9%			
(Natarajan *et al.*, 2023)			99.29%		
(Zhao *et al.*, 2022)					99.39%
(Abdulhamza & Al-Janabi, 2022)			99.66%		
(Zhang *et al*., 2022)			96%		
(Roy *et al*., 2022)			98.33%		
Our model	SXCM1				99.63%
	DPU-WVD				99.99%

The comparison table shows various models' accuracy rates for detecting specific web security vulnerabilities, including XSS (Cross-Site Scripting), SQL Injection, and Command Injection. Among these, the model by Kakisim (2024) demonstrates the highest SQL Injection detection rate at 99.96%, closely followed by Abhishek *et al.* (2023) with 99.9% for XSS detection. The models by Younas *et al.* (2024) and Roy *et al.* (2022) also focus on XSS, achieving high accuracies of 99% and 98.33%, respectively. Meanwhile, Zhao et al. (2022) and our model's DPU-WVD configuration display exceptional performance in overall detection, with 99.39% and 99.99%, respectively. Overall, the proposed model in the table achieves near-perfect accuracy with DPU-WVD at 99.99%, surpassing most other models for combined vulnerability detection, suggesting high reliability across all attack types.

CONCLUSIONS

This research paper presents an approach that integrates CNN, GRUs and an attention mechanism to identify vulnerabilities, in code written by programmers for the web development domain. Additionally showcased is the DPU-WVD dataset featuring web payloads paired with one million code snippets generated by artificial intelligence. Through training, on both the SXCM1 and DPU-WVD datasets the model achieved an accuracy rate of 99.99%. The results show that this combination model is highly effective at recognizing three vulnerability categories and surpasses cutting-edge models that usually specialize in just one type of vulnerability detection. The model shows promising potential for real world use in software development tasks with an emphasis on identifying the security leaks due to its accuracy. To guarantee the models flexibility and trustworthiness across programming scenarios and settings it is crucial to assess its performance, in diverse programming languages and environments. Future research may focus on developing real-time detection capabilities within widely used integrated development environments (IDEs), offering developers immediate feedback during the coding process. This represents a significant opportunity for additional research.

REFERENCES

Abdulhamza, F. R., & Al-Janabi, R. J. S. (2022). SQL Injection Detection Using 2D-Convolutional Neural Networks (2D-CNN). 2022 International Conference on Data Science and Intelligent Computing, ICDSIC 2022, 212–217. https://doi.org/10.1109/ICDSIC56987.2022.10075777

Abhishek, S., Ravindran, R., Anjali, T., & Shriamrut. (2023). AI-Driven Deep Structured Learning for Cross-Site Scripting Attacks. International Conference on Innovative Data

Communication Technologies and Application, ICIDCA 2023 - Proceedings, 701–709. https://doi.org/10.1109/ICIDCA56705.2023.10099960

Alarfaj, F. K., & Khan, N. A. (2023). Enhancing the Performance of SQL Injection Attack Detection through Probabilistic Neural Networks. Applied Sciences (Switzerland), 13(7). https://doi.org/10.3390/APP13074365

Arasteh, B., Aghaei, B., Farzad, B., Arasteh, K., Kiani, F., & Torkamanian-Afshar, M. (2024). Detecting SQL injection attacks by binary gray wolf optimizer and machine learning algorithms. Neural Computing and Applications, 36(12), 6771–6792. https://doi.org/10.1007/S00521-024-09429-Z

Ashlam, A. A., Badii, A., & Stahl, F. (2022). Multi-Phase Algorithmic Framework to Prevent SQL Injection Attacks using Improved Machine learning and Deep learning to Enhance Database security in Real-time. Proceedings of the 2022 15th IEEE International Conference on Security of Information and Networks, SIN 2022. https://doi.org/10.1109/SIN56466.2022.9970504

Blind SQL Injection | OWASP Foundation. (2023). https://owasp.org/www-community/attacks/Blind_SQL_Injection

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.

Command Injection | OWASP Foundation. (2021). https://owasp.org/www-community/attacks/Command_Injection

Cross Site Scripting (XSS) | OWASP Foundation. (2022). https://owasp.org/www-community/attacks/xss/

Demilie, W. B., & Deriba, F. G. (2022). Detection and prevention of SQLI attacks and developing compressive framework using machine learning and hybrid techniques. Journal of Big Data, 9(1). https://doi.org/10.1186/S40537-022-00678-0

Kakisim, A. G. (2024). A deep learning approach based on multi-view consensus for SQL injection detection. International Journal of Information Security, 23(2), 1541–1556. https://doi.org/10.1007/S10207-023-00791-Y

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space.

Mondal, B., Banerjee, A., & Gupta, S. (2022). XSS filter evasion using reinforcement learning to assist cross-site scripting testing. International Journal of Health Sciences, 11779–11793. https://doi.org/10.53730/IJHS.V6NS2.8167

Mondal, B., Banerjee, A., & Gupta, S. (2023). XSS Filter detection using Trust Region Policy Optimization. 1st International Conference in Advanced Innovation on Smart City, ICAISC 2023 - Proceedings. https://doi.org/10.1109/ICAISC56366.2023.10085076

Natarajan, Y., Karthikeyan, B., Wadhwa, G., Srinivasan, S. A., & Akilesh, A. S. P. (2023). A Deep Learning Based Natural Language Processing Approach for Detecting SQL Injection Attack. Lecture Notes in Networks and Systems, 715 LNNS, 396–406. https://doi.org/10.1007/978-3-031-35507-3_38

Nilavarasan, G. S., & Balachander, T. (2023). XSS Attack Detection using Convolution Neural Network. Proceedings of the International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering, ICECONF 2023. https://doi.org/10.1109/ICECONF57129.2023.10083807

OWASP Top Ten | OWASP Foundation. (2021). https://owasp.org/www-project-top-ten/

payloadbox/sql-injection-payload-list: SQL Injection Payload List. (2021). https://github.com/payloadbox/sql-injection-payload-list

Roy, P., Kumar, R. & Rani, P. (2022). SQL Injection Attack Detection by Machine Learning Classifier. Proceedings - International Conference on Applied Artificial Intelligence and Computing, ICAAIC 2022, 394–400. https://doi.org/10.1109/ICAAIC53929.2022.9792964

Sethi, M., Verma, J., Snehi, M., Baggan, V., Virender, & Chhabra, G. (2023). Web Server Security Solution for Detecting Cross-site Scripting Attacks in Real-time Using Deep Learning. 2023 International Conference on Artificial Intelligence and Applications, ICAIA 2023 and Alliance Technology Conference, ATCON-1 2023 - Proceeding. https://doi.org/10.1109/ICAIA57370.2023.10169255

SQL Injection | OWASP Foundation. (2023). https://owasp.org/www-community/attacks/SQL_Injection

SQLi XSS dataset. (2023). https://www.kaggle.com/datasets/alextrinity/sqli-xss-dataset

Sun, H., Du, Y., & Li, Q. (2023). Deep Learning-Based Detection Technology for SQL Injection Research and Implementation. Applied Sciences (Switzerland), 13(16). https://doi.org/10.3390/APP13169466

Tadhani, J. R., Vekariya, V., Sorathiya, V., Alshathri, S., & El-Shafai, W. (2024). Securing web applications against XSS and SQLi attacks using a novel deep learning approach. Scientific Reports, 14(1). https://doi.org/10.1038/S41598-023-48845-4

Tan, X., Xu, Y., Wu, T., & Li, B. (2023). Detection of Reflected XSS Vulnerabilities Based on Paths-Attention Method. Applied Sciences (Switzerland), 13(13). https://doi.org/10.3390/APP13137895

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need.

Yan, H., Feng, L., Yu, Y., Liao, W., Feng, L., Zhang, J., Liu, D., Zou, Y., Liu, C., Qu, L., & Zhang, X. (2022). Cross-site scripting attack detection based on a modified convolution neural network. Frontiers in Computational Neuroscience, 16. https://doi.org/10.3389/FNCOM.2022.981739

Younas, F., Raza, A., Thalji, N., Abualigah, L., Zitar, R. A., & Jia, H. (2024). An efficient artificial intelligence approach for early detection of cross-site scripting attacks. Decision Analytics Journal, 11. https://doi.org/10.1016/J.DAJOUR.2024.100466

Zhang, W., Li, Y., Li, X., Shao, M., Mi, Y., Zhang, H., & Zhi, G. (2022). Deep Neural Network-Based SQL Injection Detection Method. Security and Communication Networks, 2022. https://doi.org/10.1155/2022/4836289

Zhao, C., Si, S., Tu, T., Shi, Y., & Qin, S. (2022). Deep-Learning Based Injection Attacks Detection Method for HTTP. Mathematics, 10(16). https://doi.org/10.3390/MATH10162914

Niu, Q. and Li, X. (2020)"A High-performance Web Attack Detection Method based on CNN-GRU Model," IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 2020, pp. 804-808, doi: 10.1109/ITNEC48623.2020.9085028

Jiang, Y., Jia, M., Zhang, B. and Deng, L. (2021) "Malicious Domain Name Detection Model Based on CNN-GRU-Attention, 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 2021, pp. 1602-1607, doi: 10.1109/CCDC52312.2021.960237.