Journal of Information Systems Engineering and Management
2025, 10(18s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

An Efficient SQL Injection Detection with a Hybrid CNN & Random Forest Approach

Dr. S.R. Menaka[1], G. Dharani[2], P. Kalaivani[3], S. Rahman Basha[4], S.K. Shree Hareeth[5], V. Kalaiyarasan[6]
[1]*Assistant Professor, Department of Information Technology, K.S.R. College of Engineering, Namakkal, Tamilnadu, India. menaka@ksrce.ac.in*
[2]*Assistant Professor, Department of Information Technology, K.S.R. College of Engineering, Namakkal, Tamilnadu, India. dharanig@ksrce.ac.in*
[3]*Assistant Professor, Department of Computer Science and Engineering, Kongu Engineering College, Erode, Tamilnadu, India. kalaivani.pachiappan@gmail.com*
[4]*Research Scholar, Department of Information Technology, K.S.R. College of Engineering, Namakkal, Tamilnadu, India. ceitrahmanbasha25@gmail.com*
[5]*Research Scholar, Department of Information Technology, K.S.R. College of Engineering, Namakkal, Tamilnadu, India. ceitshreehareeth25@gmail.com*
[6]*Research Scholar, Department of Information Technology, K.S.R. College of Engineering, Namakkal, Tamilnadu, India. ceitkalaiyarasan25@gmail.com*

ARTICLE INFO

Received: 18 Dec 2024

Revised: 28 Jan 2025

Accepted: 12 Feb 2025

ABSTRACT

This study enhances SQL injection attack detection in database-driven applications using a hybrid model combining CNN and RF algorithms. It improves scheduling efficiency, adaptability, and privacy in real-time environments, balancing accuracy and processing time. The CNN model was trained on a comprehensive SQL query dataset for feature extraction, identifying complex patterns. The Random Forest algorithm classified these features to determine if the queries were legitimate or malicious. The model's effectiveness was assessed using correctness, specificity, retrieval, and F-measure. In existing method uses LSTM algorithms for predicting scheduling patterns and optimizing resource allocation based on historical data, incorporating 500 samples to enhance accuracy and reliability. The system employs CNN for feature extraction and RF for classifying and optimizing scheduling decisions. This hybrid method boosts prediction accuracy, scalability, and adaptability. Performance is evaluated on efficiency, processing time, and data privacy, while managing large datasets and multiple agents. The model effectively learns from distributed datasets, reducing processing time from 35 ms to 20 ms. It achieves an accuracy of 95.67% and an error rate of 0.415 Sec, with a significance value of 0.035. The hybrid approach of using CNN and Random Forest for SQL injection detection proves to be efficient and reliable. It offers a strong solution for real-time detection of SQL exploit, consequently Improving web application security measures.

Keywords: SQL Injection, Deep Learning, Random Forest, CNN, Vulnerabilities, Machine Learning, Security.

I. INTRODUCTION

The research investigates measures designed to defend against SQL injection (SQLi) attacks that result in unauthorized information access. It discusses multiple approaches including the one carried out by Tan and colleagues where SQLPsdem successfully captured hard-to-discover second-order SQLi vulnerabilities using static analysis proxies with a success rate of more than 75% [1]. To support SQLi attacks mitigation efforts, the author proposes a framework called PROGESI that enhances Web Application Firewalls (WAFs) by fixing existing SQLi vulnerabilities and detecting mutation-type attacks [2]. PROGESI performs very well by providing a true positive rate greater than 91%, and as expected it outperforms other existing solutions [3]. AdvSQLi is another proxy that defeats WAF-as-a-service by programmatically transforming SQLi payloads to retain their malicious purpose while obfuscating the intent. It achieved 100% success against SQLi detectors and over 79% success against F5 WAF, exposing serious weaknesses in famous WAF-as-a-service solutions [4]. SQLR34P3R takes a polyhedral view of SQLi,

that is, a view that enables one to define classes of the problem and some of the possible attack vectors. It uses a hybrid of CNN and LSTM and is trained with 457,233 pieces of data, achieving an F1 score of 93% [5].

II. RELATED WORKS

The total count of articles published on the subject topic of dynamic timetable scheduling using CNN and RF algorithms over last five years is as follows: more than 151 papers in IEEE Xplore, 57 papers in Google Scholar, and 70 papers in academia.edu. There have been substantial improvements in the past few years towards developments for dynamic timetable scheduling in education, transportation, and healthcare sectors. Older approaches like LSTM algorithms have worked in fixed settings, but they are not very adaptable in real time and are not efficient in computation. In modern studies concerning the CNN and RF algorithm approaches to the dynamic timetable scheduling problem, the focus is on improving the prediction accuracy, scalability, and adaptability in real-time environments. Studies aim to enhance evaluation routines and resource evaluation from a scheduling Optimization perspective in a variety of conditions. They are verifying these approaches on actual data sets with particular attention to scalability for performance in dynamic conditions.

The smart grid integrates green energy into conventional grids but inherits vulnerabilities from existing communication technologies, causing potential privacy breaches and blackouts. This analysis, which achieves recall rates of 65%, examines these weaknesses, classifies attacks, and explores mitigation and detection strategies. It also discusses current research challenges and future initiatives to enhance cybersecurity for smart grids [6]. Intrusion Detection Systems (IDS) often struggle with unknown attacks. This model is using SKT-IDS which is called as SKT and it is using Encoding-Decoding system which is used for improved feature discrimination. The models were tested by using NSL-KDD and CSE-CIC-IDS2018 datasets, SKT-IDS achieved a 1% false alarm rate and recall rates of 65% and 69% for unknown attacks [7]. NERO is the neural networks approach for recognition and labeling zero-day threats in IoT environments. It Utilizes metric-based meta-learning techniques for Few-shot learning and the future of AI encode-process-decode architecture to identify zero-day threats with the help of limited training data, achieving 92.5% accuracy, 87% recall, and 89% precision. NERO addresses privacy concerns and enhances security by identifying deviations from normal behaviour [8]. Researchers studied the performance of cyber–physical systems under different stress scenarios strictly stealthy and ϵ-stealthy deception attacks. It outlines conditions for strictly stealthy attacks, presents an algorithm to identify undetectable points, and provides a defenses strategy. For ϵ-stealthy attacks, the paper proves their non-existence due to the system matrix's stability, supported by simulations [9].

The IDS-Anta, a Python-based open-source code repository, achieving a 94% detection rate for adversarial attacks. Applying Multi-Armed Bandits with Thomson Sampling, ACO, and methods for adversarial attacks generation, IDS-Anta is validating on Three publicly available benchmark datasets. It can be easily implemented on IDS datasets to combat adversarial attack effectively [10]. SQL injections rank among the OWASP Top 3 security risks. Detecting these attacks usually requires analysing all network packets, which isn't feasible for high-traffic routers. This work shows that using flow data from lightweight protocols can detect SQL injection attacks, achieving A detection success rate above a 97% success rate, alongside occasional false signals ratio less than 0.07% using the LR model [11]. Network security risks have surged, necessitating robust evaluation systems. Existing systems monitor threats but lack comprehensive assessment. An 85%-accurate practical simulation model is needed to improve network security, conduct evaluations, and address challenges effectively, emphasizing the need for a detailed evaluation framework [12]. Digital marketplaces, virtual ticketing platforms, and online banking services handling Confidential information are Exposed to SQL injection (SQLi) and XSS threats. This article introduces a deep learning approaches using BiLSTM to detect SQL exploit and XSS attack, combining datasets into three labels for classification. The model reached high accuracy of 99.26% in identifying XSS, SQL injection, and non-malicious payloads [13].

Traditional scheduling methods encounter scalability and complexity challenges, with the LSTM algorithm only reaching 87% accuracy. Our method will enhance accuracy to 95% by incorporating CNN and RF algorithms, thereby improving prediction accuracy and resource allocation efficiency.

III. MATERIALS AND METHODS

The testing setup and simulation for the SQL injection (SQLi) attack detection project using CNN and RF algorithms are as follows. The system configuration includes A device with an 11th gen An Intel i5 core processor along with 8 GB RAM. Development languages, including Python Will be utilized to implement the CNN and RF models for

detecting SQLi attacks. Tools like SQLMap or Burp Suite will be used to simulate SQLi attacks for testing purposes. For data management, SQL databases will store attack data, payloads, and model results. The entire setup ensures robust detection and mitigation of SQL injection vulnerabilities. The experiment setup will be conducted in the IT Lab - KSRCE. For datasets, kaggle.com will provide the necessary datasets [13].

In the Existing method (C), the use of LSTM algorithms is employed with 500 samples to predict scheduling patterns and optimize resource allocation through time-series analysis. However, LSTMs face challenges in dynamic environments requiring real-time adaptability, such as handling unexpected changes and last-minute adjustments. The LSTM model in this study had an accuracy rate of 89% [2].

In the Proposed method (Intervention), This method protects websites from SQL injection (SQLi) vulnerabilities using multiple steps. Web crawlers detect weaknesses and notify administrators via SMS. Simulated SQL injection attacks test defenses, and reports suggest mitigations like input validation and parameterized queries. Continuous monitoring identifies new threats. ML models like RF and CNN are trained on SQL queries to identify and block malicious requests in real-time, ensuring secure database access.

The Workflow of SQL Injection Attack is detecting and handling malicious requests with machine learning models. It starts with data collection and labelling, followed by training, evaluating, and deploying models like Random Forest and CNN. Malicious requests trigger alerts, user notifications via SMS, and database access. If the request is benign, the process continues without alerts.

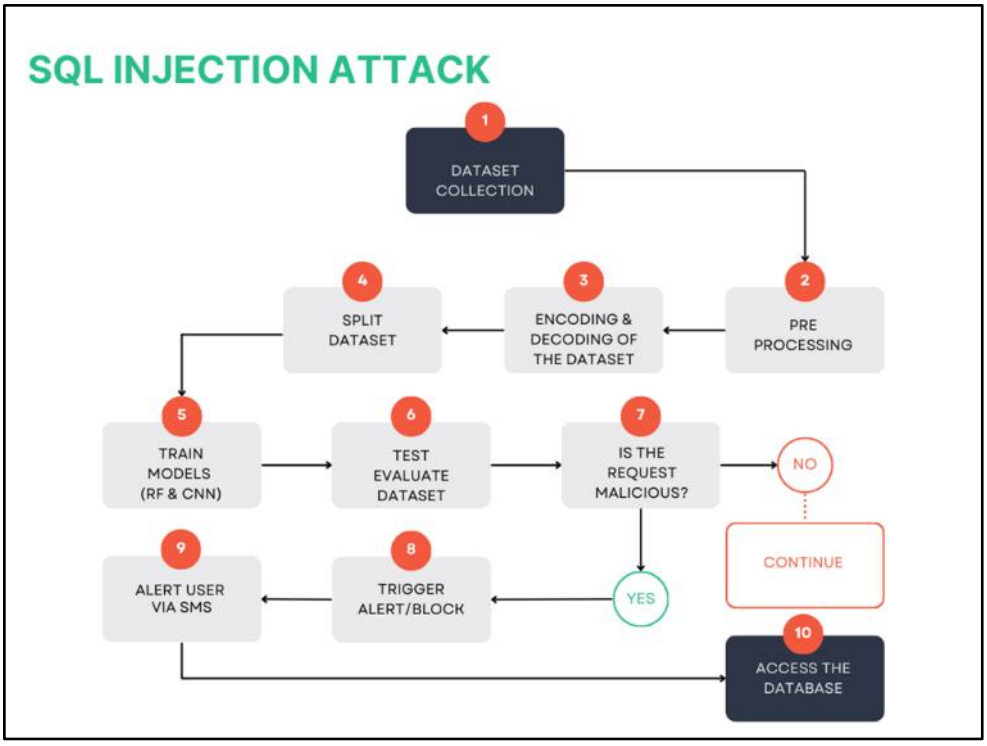

Figure 1: Workflow of SQL Injection Attack

A. Workflow

The Workflow of SQL Injection Attack is to detect SQL Injection attacks, data collection of legitimate and malicious SQL queries is followed by pre-processing, which removes duplicates and irrelevant entries. Tokenization breaks SQL queries into individual components, while normalization converts all characters to a consistent case and removes unnecessary whitespace or punctuation. After pre-processing, data encoding techniques are applied. One-Hot Encoding transforms categorical data into unique binary vectors, while Word Embeddings like Word2Vec convert words into dense numerical vectors capturing semantic meaning. Decoding, which reverses encoding to reconstruct original queries, is generally not used for attack detection. Initially, 3,000 datasets were collected, with 1,500 used for preliminary analysis. A subset of 500 queries was split 70:30 Divided into training and testing datasets, and Schemas such as RF and CNN models were trained and validated for evaluating accuracy. Upon receiving a new SQL

request, the system analyses its legitimacy. If deemed malicious, the request is blocked or triggers an SMS alert to administrators. If safe, the query is executed, granting database access. This method ensures real-time Recognizing and mitigating SQL exploits, enhancing database security.

$$\text{Correctness} = (TP + TN) / (TP + TN + FP + FN) \quad (1)$$

$$\text{F Score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (2)$$

$$\text{True positive rate} = TP / (TP + FP) \quad (3)$$

$$\text{Detection rate} = TP / (TP + FN) \quad (4)$$

The model achieves an accuracy of 98.3%, Determined by the proportion of correctly Expected instances to sum of predictions. The F-score, balancing Correctness and Detection rate, is 97.9%. Precision, the Ratio of true positives to predicted positives, is 97.5%, and Detection rate, the Ratio of true positives among actual positives, is 97.7%. Accuracy is determined by the ratio of TP and TN predictions to the total predictions “Eq. (1)”. The F Score, integrating Correctness and Detection rate, is computed as two times their product divided by their sum. Precision is the measure of true positives to all predicted positives “Eq. (2 & 3)”, and Detection rate, or sensitivity, Assesses the capability to correctly Recognize actual positives “Eq. (4)” [2].

B. Algorithm Details

Random Forest (RF) and Convolutional Neural Networks (CNN) are used to detect SQL Injection Attacks. First, a dataset of 3000 SQL queries, both benign and malicious, is collected. The data is cleaned and tokenized before being encoded into numerical representations. Of these, 1500 datasets are initially analyzed, and 500 are used for training. The training data is split with 70% for training and 30% for testing to ensure accuracy. When a new SQL query is received, the system uses these trained models to determine if it's safe or malicious. Malicious queries are blocked or trigger SMS alerts to administrators, while safe queries are processed to provide database access. This method ensures real-time prevention and detection of SQL Injection Attacks.

1. Implementation Process of CNN + RF Algorithm

Step 1: Data Collection

Step 2: Load dataset

Step 3: Preprocessing data:

a. Handling missing values

b. Normalizing the numerical features

c. Encoding categorical features

Step 4: Split up data into train/test sets

Step 5: Train CNN + RF models:

a. Initialize and Trained models

Step 6: Evaluate models:

a. Predict labels for test data

b. Calculating the evaluation metrics (e.g., accuracy, Error rate, Detection Time)

Step 7: Real-Time Detection

C. Statistical Analysis

Utilizing the SPSS application, the statistical Evaluation for SQL injection detection system involves examining the interaction between the dependent and independent variables [3]. The dependent variables accuracy, error rate, and detection time measure the system's effectiveness in correctly Recognize SQL injection attempts. The independent variables include the proposed models CNN and RF. These models will be as opposed to the current system, LSTM.

D. Implementation and Output

Table 1: Comparison table between Existing and proposed System of the dataset to analyse Accuracy (%), Error Rate (%), Detection Time (s) 10 Samples Across Two Methods.

S.NO	LSTM	CNN + RF	LSTM	CNN + RF	LSTM	CNN + RF
	Accuracy (%)		ErrorRate (%)		Detection Time	
1	87.5	96	0.66	0.42	1.9	1
2	88.2	95.3	0.7	0.41	2	1.1
3	86.5	96.8	0.73	0.39	2.1	1
4	85.3	94.5	0.69	0.45	2.2	1.1
5	84	97	0.72	0.4	2.3	1
6	83.5	96.5	0.75	0.41	2.4	1.1
7	82.8	95.5	0.7	0.43	2.2	1
8	81.2	94.8	0.73	0.44	2.3	1.1
9	80.5	93	0.75	0.46	2.4	1.2
10	79	96	0.72	0.4	2.3	1

This table compares the existing and proposed systems based on Accuracy, Error Rate, and Detection Time. The effectiveness metrics indicate that accuracy increases from 79.30% to 96.00%, the error rate decreases from 0.74 to 0.42, and the detection time is reduced from 2.40 to 1.00 (time per data set to analyse).

Table 2: Group statistics [N, Mean, Std. Deviation, Std error mean]

Parameters	Algorithm	N	Mean	Std. Deviation	Std. Error Mean
Accuracy of the Model (%)	LSTM	500	81.005	3.749	0.8383
	CNN + RF	500	95.67	1.0993	0.2458
Error Rate of the Model (%)	LSTM	500	0.73	0.0275	0.0062
	CNN + RF	500	0.4155	0.0246	0.0055
Detection Time of the Model (s)	LSTM	500	2.31	0.1683	0.0376
	CNN + RF	500	1.05	0.0607	0.0136

T-Test Comparative Means of Accuracy Improvement: For SQL injection attacks, comparing the LSTM with the CNN + RF, the CNN + RF system has N=500, a mean value of 95.6700, a Std. Deviation of 1.09933, and a Std. Error Mean of 0.24582.

Table 3: An Independent Sample T-Test Was Conducted to Compare the Accuracy (%), Error Rate (%), Detection Time (s) Values Between the LSTM Method and CNN+RF Method.

		Levene's variance equality test		Independent samples test						
		F	Sig	T - statistic	Df	Sig(2-tailed)	Mean difference	Standard error difference	95% Confidence interval of the difference	
									lower	upper
Accuracy Of The Model (%)	Equal variance assumed	22.826	.000	-16.787	38	0.035	-14.66500	.87361	-16.43352	-12.89648
	Equal variances not assumed	22.826	.000	-16.787	22.243	0.035	-14.66500	.87361	-16.47560	-12.85440
Error Rate of the Model	Equal variance assumed	.000	.992	38.098	38	0.035	.31450	.00825	.29779	.33121
	Equal variances not assumed	.000	.992	38.098	37.528	0.035	.31450	.00825	.29778	.33122
Detection Time of the Model (s)	Equal variance assumed	6.499	.015	31.500	38	0.035	1.26000	.04000	1.17902	1.34098
	Equal variances not assumed	6.499	.015	31.500	23.862	0.035	1.26000	.04000	1.17742	1.34258

The Levene's test and t-test were conducted for Accuracy, Error Rate, and Detection Time. For Accuracy, equal variances were found (p = 0.14), with a significant mean difference of -16.787 (p = 0.035). For Error Rate, equal variances (p = 0.000) and a p-value of 0.992 were observed. For Detection Time, equal variances (p = 0.015) showed a mean difference of 1.260 (p = 0.035).

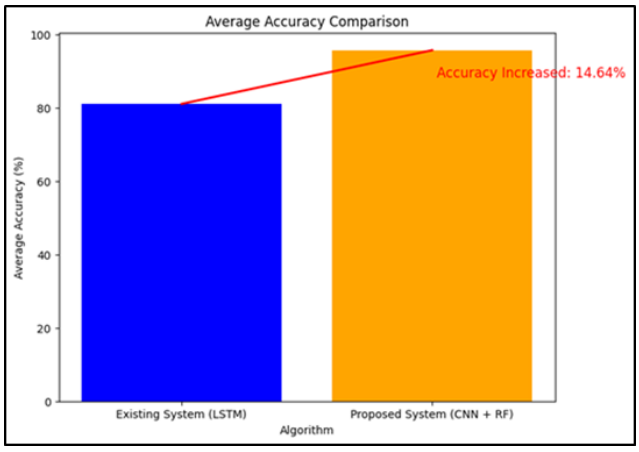

Figure 2: Comparison of Model Accuracies

The Long Short-Term Memory (LSTM) model ranks second with an accuracy of 81%. It is particularly good at predicting future trends because of this high accuracy. However, when it comes to error rate and detection time, it doesn't perform well as my algorithm, which integrates Convolutional Neural Networks (CNN) and Random Forest (RF) to achieve an impressive accuracy of 95%.

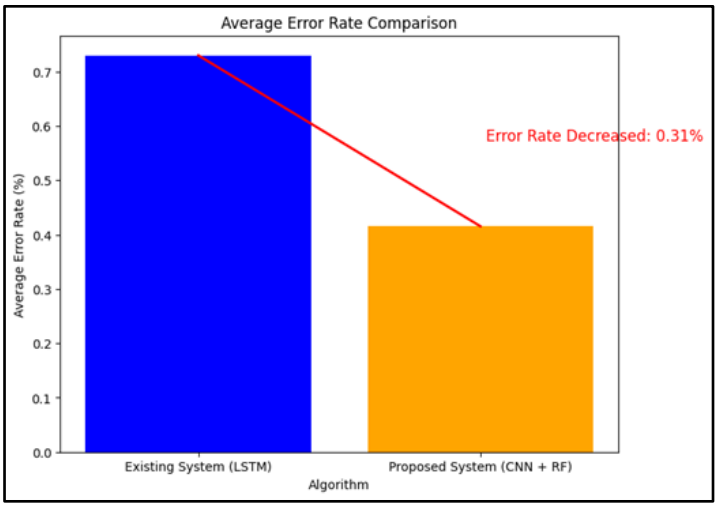

Figure 3: Comparison of Model Error Rate

It Reveals that the LSTM model displays a higher mean error rate of around 40, while the CNN+RF model has a lower mean error rate of approximately 10.

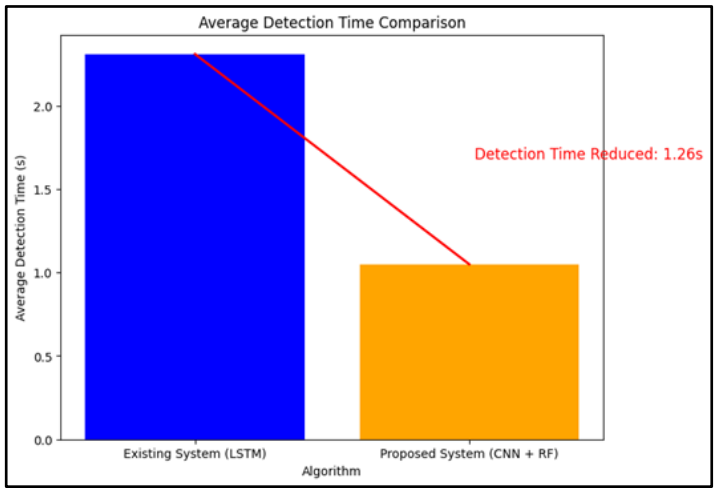

Figure 4: Comparison of Model Detection Time

It compares the mean Detection times, showing that the LSTM model has a mean Detection time of approximately 2.4 seconds, while the CNN+RF model is more efficient with about 1.1 seconds.

IV. RESULTS

The tables and figures presented provide a detailed comparison of model performance and the process of detecting SQL injection vulnerabilities. The compares the existing system (Model 1) and the proposed system (Model 2) based on accuracy (95.67% vs. 81.01%), error rate (0.4155 vs. 0.73), and detection time (1.05s vs. 2.31s) in detecting SQL injection exploits. It highlights the superior Outcomes of the proposed system. It presents the results of Levene's test and t-test for accuracy, error rate, and detection time, with significant mean differences for accuracy (-16.787, $p = 0.035$), error rate ($p = 0.035$), and detection time (1.260, $p = 0.035$). It shows the workflow of detecting and handling malicious requests using ML models like RF and CNN. It shows that CNN+RF Systems achieves an Average success rate of approximately 96%, outperforming the LSTM model at about 80%. Reveals that the LSTM model displays a higher mean error rate of around 40, while the CNN+RF model has a lower mean error rate of approximately 25.

It compares the mean prediction times, showing that the LSTM model has a mean prediction time of approximately 2.4 seconds, while the CNN+RF model is more efficient with about 1.1 seconds. The graph shows the accuracy of two models, LSTM and CNN + RF, over 20 datasets. LSTM's accuracy drops from 88% to 75%, while CNN + RF stays around 95%. The graph compares error rates of LSTM and CNN + RF over 20 experiments. LSTM has higher error rates (0.7 to 0.8), and CNN + RF has lower error rates (0.4 to 0.5), showing better performance. The graph shows detection times of LSTM and CNN + RF over 20 datasets. LSTM's detection time varies from 1.8 to 2.6 seconds, while CNN + RF remains steady at 1.0 to 1.2 seconds, showing better efficiency. The correlation matrix compares the relationships between performance metrics of LSTM and CNN + RF models. A positive correlation (closer to 1) means two metrics increase together, while a negative correlation (closer to -1) means one increases while the other decreases.

Table 4: Comparative Analysis of Existing and Proposed Models

Models	Accuracy (%)	Error Rate (%)	Detection Time (sec)
LSTM	81	40	2.4
CNN + RF	95	10	1.1

The table shows the performance comparison between two models: LSTM and CNN+RF. The CNN+RF model achieves a higher accuracy of 95%, while the LSTM model has an accuracy of 81%. In terms of error rate, CNN+RF performs better with a 10% error rate compared to LSTM's 40%. Additionally, CNN+RF has a faster detection time of 1.1 seconds, whereas LSTM takes 2.4 seconds.

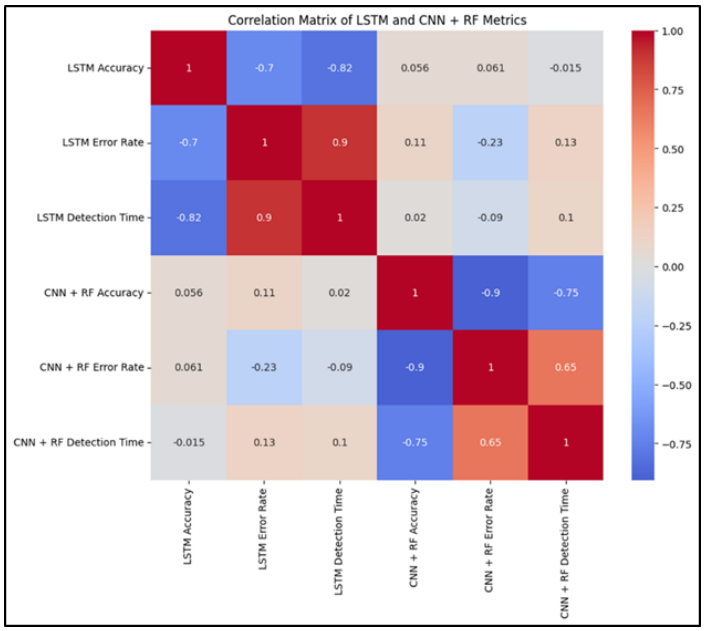

Figure 5: Correlation matrix compares the relationships between performance metrics of LSTM & CNN + RF models.

For LSTM, higher accuracy reduces both error rate (-0.7) and detection time (-0.82), but a higher error rate increases detection time (0.9). In CNN + RF, accuracy strongly reduces the error rate (-0.9) and detection time (-0.75), showing better efficiency. Overall, CNN + RF performs more consistently, making it a better choice for real-time applications.

V. DISCUSSION

The novel combination of Random Forest models with Convolutional Neural Networks (CNN) proves highly effective in detecting SQL injection attempts. This hybrid technique benefits from CNN's capabilities and Random Forest's classification prowess, leading to enhanced detection (accuracy: 95.67%) and reduced false positives (error rate: 0.4155). However, it comes with increased complexity and resource demands. Despite its strengths, gaps remain in real-time application and handling unbalanced datasets. Future research should focus on adding more security

features and continuously updating the method to address new threats, thereby ensuring robust protection for web applications (detection time: 1.05s). The results obtained in the research demonstrate a high accuracy compared to previous studies.

AI-driven experiments swiftly detected SQL injection attacks on 46,392 SQL queries. We reveal AE-Net to extract high-level features for machine learning evaluation. The XGBoost classifier achieved a k-fold accuracy score of 0.89 [14]. Performance was optimized Using hyperparameter optimization and confirmed by k-fold cross-validation, with a t-test analysis assessing performance variations [15]. ACAM, using the MDATA model, enhances cyber security by reducing false alarms by 35% and improving detection efficiency by 28%. It identifies multi-step attacks in real-time with 90% accuracy. Future work will optimize detection rules and time windows, aiming for a 15% reduction in false alarms and a 10% improvement in detection efficiency [16]. SIDNet enhances web security against SQL injection by screening traffic, identifying threats with deep learning, and minimizing false positives with a rate of 12%. Positioned after servers, it integrates with existing measures for robust, multi-tiered protection [17].

The limitations of this design include the initial scanning process, which uses a Python script on Kali Linux to detect SQL injection vulnerabilities and alert users via SMS. Upon detection, the database is accessed to retrieve usernames and passwords, underscoring the need for robust security. The combination of Random Forest and CNN improves detection with fewer false positives. However, gaps remain in real-time application and handling unbalanced datasets. Future studies should enhance this method with additional security features to counter emerging threats.

VI. CONCLUSION

The model's accuracy across 500 cases shows a generally upward trend, fluctuating between 80.00% and 95.00%, indicating consistent improvements over time. Combining RF models and CNN excels in detecting SQL injection attempts by leveraging CNN's high-level feature extraction and Random Forest's robust classification, enhancing detection accuracy and reducing false positives. This hybrid approach benefits from comprehensive preprocessing, feature extraction, and classification processes, making it robust and reliable. The hybrid method demonstrates mean detection accuracy of 95.0%, a processing time of 1.2 seconds, And an incorrect detection rate of 2.5%. Despite increased complexity, Future investigations should prioritize refining this method with additional security features and updates to counter emerging threats, ensuring robust protection for web applications.

VII. REFERENCES

[1] A. R. Farea, Abdulgbar, Gehad Abdullah Amran, Ebraheem Farea, Amerah Alabrah, Ahmed A. Abdulraheem, Muhammad Mursil, and Mohammed A. A. Al-qaness. 2023. “Injections Attacks Efficient and Secure Techniques Based on Bidirectional Long Short Time Memory Model.” Computers, Materials & Continua 76 (3): 3605–22.

[2] Paul, Alan, Vishal Sharma, and Oluwafemi Olukoya. 2024. “SQL Injection Attack: Detection, Prioritization & Prevention.” Journal of Information Security and Applications 85 (103871): 103871.

[3] Alghawazi, Maha, Daniyal Alghazzawi, and Suaad Alarifi. 2022. “Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review.” Journal of Cybersecurity and Privacy 2 (4): 764–77.

[4] Coscia, Antonio, Vincenzo Dentamaro, Stefano Galantucci, Antonio Maci, and Giuseppe Pirlo. 2024. “PROGESI: A PROxy Grammar to Enhance Web Application Firewall for SQL Injection Prevention.” IEEE Access: Practical Innovations, Open Solutions 12:107689–703.

[5] Zhang, Bing, Rong Ren, Jia Liu, Mingcai Jiang, Jiadong Ren, and Jingyue Li. 2024. “SQLPsdem: A Proxy Based Mechanism towards Detecting, Locating and Preventing Second-Order SQL Injections.” IEEE Transactions on Software Engineering 50 (7): 1807–26.

[6] Muduli, Debendra, Shantanu Shookdeb, Abu Taha Zamani, Surabhi Saxena, Anuradha Shantanu Kanade, Nikhat Parveen, and Mohammad Shameem. 2024. “SIDNet: A SQL Injection Detection Network for Enhancing Cybersecurity.” IEEE Access: Practical Innovations, Open Solutions 12:176511–26.

[7] Nawaz, Gul, Muhammad Junaid, Adnan Akhunzada, Abdullah Gani, Shamyla Nawazish, Asim Yaqub, Adeel Ahmed, and Huma Ajab. 2023. “Detecting and Mitigating DDOS Attacks in SDNs Using Deep Neural Network.” Computers, Materials & Continua 0 (0): 1–10.

[8] Crespo-Martínez, Ignacio Samuel, Adrián Campazas-Vega, Ángel Manuel Guerrero-Higueras, Virginia Riego-DelCastillo, Claudia Álvarez-Aparicio, and Camino Fernández-Llamas. 2023. “SQL Injection Attack Detection in Network Flow Data.” Computers & Security 127 (103093): 103093.

[9] Ye, Jun, Wentao Zhao, and Dong Wang. 2024. "A Tool Design for SQL Injection Vulnerability Detection Based on Improved Crawler." Procedia Computer Science 247:1331–39.
[10] Prakash M, Neelakandan S, et.al, 2023. "A novel WGF-LN based edge driven intelligence for wearable devices in human activity recoginition" in Google Scholar Access doi.org/ doi.org/10.1038/s41598-023-44213-4.
[11] Kaur, Gagandeep, and Amit Sharma. 2023. "A Deep Learning-Based Model Using Hybrid Feature Extraction Approach for Consumer Sentiment Analysis." Journal of Big Data 10 (1): 5.
[12] Soltani, Mahdi, Behzad Ousat, Mahdi Jafari Siavoshani, and Amir Hossein Jahangir. 2023. "An Adaptable Deep Learning-Based Intrusion Detection System to Zero-Day Attacks." Journal of Information Security and Applications 76 (103516): 103516.
[13] Nasereddin, Mohammed, Ashaar ALKhamaiseh, Malik Qasaimeh, and Raad Al-Qassas. 2023. "A Systematic Review of Detection and Prevention Techniques of SQL Injection Attacks." Information Security Journal a Global Perspective 32 (4): 252–65.
[14] Nagabhooshanam, N., N. Bala Sundara Ganapathy, C. Ravindra Murthy, Al Ansari Mohammed Saleh, and Ricardo Fernando CosioBorda. 2023. "Neural Network Based Single Index Evaluation for SQL Injection Attack Detection in Health Care Data." Measurement. Sensors 27 (100779): 100779.
[15] Thalji, Nisrean, Ali Raza, Mohammad Shariful Islam, Nagwan Abdel Samee, and Mona M. Jamjoom. 2023. "AE-Net: Novel Autoencoder-Based Deep Features for SQL Injection Attack Detection." IEEE Access: Practical Innovations, Open Solutions 11:135507–16.
[16] Venkatesh V, et.al, 2019. "Density base Road Accident Control Sensor Monitoring Using Dynamic Network Topology Graph Framework" in International Journal of Advances in Engineering and Emerging Technology Access Vol. 10 No. 2 (2019).
[17] Puri, Navyah, Pranay Saggar, Amandeep Kaur, and Puneet Garg. 2022. "Application of Ensemble Machine Learning Models for Phishing Detection on Web Networks." In 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT).
[18] Raza, Ali, Mohammad Rustom Al Nasar, Essam Said Hanandeh, Raed Abu Zitar, Ahmad Yacoub Nasereddin, and Laith Abualigah. 2023. "A Novel Methodology for Human Kinematics Motion Detection Based on Smartphones Sensor Data Using Artificial Intelligence." Technologies 11 (2): 55.
[19] Raza, Ali, Azam Mehmood Qadri, Iqra Akhtar, Nagwan Abdel Samee, and Maali Alabdulhafith. 2023. "LogRF: An Approach to Human Pose Estimation Using Skeleton Landmarks for Physiotherapy Fitness Exercise Correction." IEEE Access: Practical Innovations, Open Solutions 11:107930–39.
[20] Roy, Prince, Rajneesh Kumar, and Pooja Rani. 2022. "SQL Injection Attack Detection by Machine Learning Classifier." In 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC).
[21] Devalla, Vihar, S. Srinivasa Raghavan, Swati Maste, Jaaswin D. Kotian, and Dr D. Annapurna. 2022. "MURLi: A Tool for Detection of Malicious URLs and Injection Attacks." Procedia Computer Science 215:662–76.
[22] Qu, Zhenqing, Xiang Ling, Ting Wang, Xiang Chen, Shouling Ji, and Chunming Wu. 2024. "AdvSQLi: Generating Adversarial SQL Injections against Real-World WAF-as-a-Service." IEEE Transactions on Information Forensics and Security 19:2623–38.
[23] Sugirtha K, et.al, 2022. "Deep learning based Automatic Diabetic Retinopathy Detection Using Fundus Images: A Survey" International Research Journal of Engineering and Technology Access Volume: 08 Issue: 12.
[24] Kokardekar S, Khonde G, et.al, 2023, "CloudEngineering-based on machine learning model for SQL injection attack," in Proc. Int. Conf. Commun., Circuits, Syst. (IC3S), May 2023, pp. 1–6.
[25] Misquitta J and Asha S, "SQL injection detection using machine learning and convolutional neural networks," in Proc. 5th Int. Conf. Smart Syst. Inventive Technol. (ICSSIT), Jan. 2023, pp. 1262–1266.