Journal of Theoretical and Applied Information Technology
15th June 2025. Vol.103. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

COLLABORATIVE SQL AND JSON INJECTION DETECTION SYSTEM USING MACHINE LEARNING

ALAA S. ALNEMARI[1] , SAMAH H. ALAJMANI [2]
Taif University, College of Computers and Information Technology, Department of Information Technology, Taif, Saudi Arabia[1 2]
E-mail: alaa_salem_alnemari@outlook.com[1], s.ajmani@tu.edu.sa [2]

ABSTRACT

SQL and JSON injection attacks are still a significant security vulnerability in contemporary web applications, particularly in API-based systems. This paper introduces a new collaborative machine learning system designed to detect and mitigate SQL and JSON injection attacks in real time. The system adopts a stratified defensive approach, integrating database query analysis, behavioural scrutiny of API requests, and instantaneous anomaly detection to establish a resilient protective framework. Utilizing advanced machine learning techniques—including Support Vector Machines (SVM), Naive Bayes (NB), Decision Trees (DT), and Random Forest (RF)—it achieves high-fidelity discrimination between benign and malicious queries. Also, the system maintains dynamic response to new attack methods through real-time threat monitoring, input sanitization mechanisms, and adaptive learning strategies. The model is trained on a mixed dataset of labelled SQL and JSON injection attempts along with actual queries, which enhances its accuracy in detection. Empirical evaluations demonstrate 94% accuracy with zero false positives compared to conventional syntax-based detection mechanisms. Future improvements may involve the application of transformer-based architectures (e.g., BERT, GPT-activated detection), graph neural networks (GNNs), and reinforcement learning to enhance accuracy and responsiveness. This research highlights the need for multi-pronged security that is AI-driven to safeguard modern database systems and API infrastructures against advanced SQL-JSON injection attacks.

Keywords: *SQL Injection, JSON Injection, Machine Learning-based Detection, API Security, Real-time Anomaly Detection,*

1. INTRODUCTION

Over the past few years, web application development has become unavoidable. This evolution and complete dependency on technology advancements pushed attackers to increase the power of their tools and identify vulnerabilities in websites and obtain specific information from databases. Amongst the most complex security flaws is the injection that enables attackers to access databases and get useful information from them or enables them to know what they contain inside them. Injections are divided into various types like SQL and JSON. SQL and JSON injection attacks, two of the most dangerous vulnerabilities in web applications, are commonly utilized by attackers to hijack unauthorized access, manipulate data, and disrupt normal business logic. Placing malicious SQL queries within input parameters, for instance, login pages or search inputs, SQL injection (SQLi) particularly targets relational databases in order to tamper with the underlying database. An attacker can bypass the authentication controls, steal sensitive data, including financial information and user credentials, and even modify or delete important data. SQL injection, in worst-case scenarios, enables hackers to execute administrative commands on the database server, leading to a complete system compromise. The ease of SQL injection and the frequently poor input validation and query sanitization provided by web applications contribute to its ongoing threat. OWASP writes that the SQL Injection (SQLi) web security vulnerability arises when an attacker injects an SQL statement into a web application by altering the client's input data. If carried out perfectly, an SQLi attack can compromise sensitive information like passwords, credit card numbers, and personal data[1]. It also offers attackers the chance to tamper with database information through actions like inserting, updating, or deleting information. Additionally, they are able to perform administrative tasks in the database like stopping the database management system and executing commands on the operating system level[1]. Even if the attacker does not know the

username or password, SQLIA is a database attack that alters SQL queries to make them return true.

But how is this accomplished? To accomplish this, utilize a logic gate called an OR logic gate. Let us first talk about the OR logic. The following are the OR logic operations B. A and b, which have two inputs and one output: The output will be false when both inputs are false. The output is true if the inputs are true. The output is true if all the inputs are true. Regardless of the rest of the inputs, the output will always be true if the inputs are true[1]. The main use of the single quote is closing a string parameter in most scenarios.

For example, if a user enters data in the input area of a web application, the same is viewed as a string. In most instances, more so when inputting usernames and passwords, a single quote is used to delimit this parameter string. The most significant section of this process is the OR function and the statement "1 = 1." This line contains two sides: the left and the right. The attention, in this case, is on the right, "1 = 1," and this will always hold because one is always equal to one. If either of the conditions in the OR function is true, then the overall statement is true. Therefore, since "1 = 1" is always true, the SQL query produces a true result [1].

Furthermore, the single quotation mark (‘’) can serve as a delimiter to truncate subsequent portions of an SQL query. This effectively nullifies any conditional checks within the statement, causing the OR operator to evaluate as true and thereby permitting unauthorized authentication—a fundamental mechanism of SQL injection (SQLI) vulnerabilities. Conversely, JSON injection attacks take advantage of weaknesses in applications that utilize JSON (JavaScript Object Notation) for inter-server and inter-client data communication. Poor handling of untrusted JSON data can lead to JSON injection, allowing an attacker to inject bad objects or code in the payload. This could lead to data leaks, arbitrary code execution, or even illicit changes in the application behaviour.

A JSON Web Token (JWT) is a standardized format for the secure exchange of cryptographically signed JSON-encoded information between distributed systems. While most commonly applied to transferring user authentication claims in session management and access control models, JWTs theoretically have the ability to encapsulate arbitrary data payloads. Unlike session tokens, JWTs contain the entire client-state data in the token itself. This attribute makes them particularly valuable in horizontally scaled web designs, where interoperability among multiple backend servers must be seamless [2].

JSON injection is especially risky in the context of NoSQL databases like MongoDB because the attacker can use operators like $gt, $ne, and $regex to create evil queries that can evade security measures, read sensitive information, or update database records. These exploits leverage the inherent flexibility of JSON's dynamic structure and the schema-less design of the common NoSQL systems, where no strict data structuring requirements exist as in relational databases. The prevalence of these vulnerabilities, along with the possibility of very high-level disruption of operation, renders both SQL and JSON injection attacks significant security threats to contemporary data systems.

Web applications today are database-dependent, and weaknesses in the way that they take in input open companies to data breach, monetary loss, government penalty, and reputation damage. Cyber attackers prefer to exploit such vulnerabilities also because of the relative ease of attacking them with automated tools or even manually creating payloads. Efficient security controls under the cover of input validation, query sanitization, and dynamic detection controls must be implemented to neutralize such attacks. The machines can learn from changing patterns of attacks through machine learning and blended detection approaches to offer real-time protection against these extremely critical vulnerabilities.

The shortcomings of current SQL and JSON injection detection systems render them ineffective to counter the growing sophistication of cyber-attacks. Conventional detection techniques, i.e., rule-based or signature-based systems, identify threats by utilizing known attack signatures or predetermined patterns. Such techniques have high false-negative rates as they fail to detect unknown or obfuscated injection techniques effectively, despite their good performance with familiar attack vectors. Moreover, because anomaly-based systems cannot distinguish between benign anomalies and real attacks, they tend to have high false-positive rates. The systems rely on deviations from normal behaviour. Moreover, most of the systems available in the market today are those that are meant to counter SQL or JSON injection attacks separately, which leaves applications using relational and non-relational databases vulnerable. The absence of hybrid technologies that can protect against SQL attacks as well as JSON injection is another significant disadvantage. A detection approach using compartmentalization cannot offer

absolute protection against the ubiquity of contemporary web applications that combine JSON-based NoSQL databases and SQL-based relational databases.

Existing solutions are patchwork, securing just a single database type or single injection method, which leaves the organization vulnerable to sophisticated attack methods that take advantage of SQL and JSON weaknesses in combination. Now, systems are able to learn and adapt dynamically to new attack behaviour in real-time thanks to machine learning, an appealing solution compared to older methods. Machine learning models, in contrast to static rule-based systems, have the ability to parse through large amounts of data and recognize subtle patterns and correlations that could represent malicious activity. While unsupervised techniques possess the potential to detect anomalies without having any prior knowledge of the attack signatures, supervised learning models enjoy the benefit of being trained with labelled malicious and benign query data.

There are, however, challenges associated with the deployment of machine learning-based detection systems. Web applications produce enormous volumes of data in real-time, and this necessitates the need for efficacious processing and model inference, thereby rendering scalability a very significant concern. Also, regular retraining of models and access to varied and current datasets are necessary for responsiveness to new attack methods. The foregoing application of machine learning-based solutions is compounded by the requirement of sacrificing detection effectiveness for system usability. Finally, machine learning offers a more complete and responsive solution than traditional detection systems, limited by their compartmentalized nature and dependence on rigid rules. In order to realize the full potential of machine learning to detect SQL and JSON injection attacks, however, scalability, responsiveness, and low false positives must be tackled. The threat of injection attacks, such as SQL injection and JSON injection, is a significant cyber security challenge since our dependency on relational and non-relational databases for mission-critical applications continues to grow. JSON injection targets the document-oriented data structures commonly used in non-relational databases, while SQL injection targets vulnerabilities of structured query languages, primarily against relational databases.

Current detection systems fail to possess the ability to ward off a number of injection attacks since they are designed specifically to address a certain category of databases or lines of attack. In addition, the need is there for a faster mechanism capable of combating unknown and new threats because patterns of attacks constantly change. Old school rule-based or signature-based detection approaches tend to fail, opening systems up to more modern and clever attack methodologies. It is therefore vitally necessary that an improved hybrid detection system that is capable of thoroughly detecting injection attacks across a vast array of database environments is created. Relational and non-relational databases need to be extensively protected by such a system, which needs to utilize machine learning models to provide dynamic and adaptive threat analysis.

In light of these challenges, this research strongly concentrates on detection methods of hybrid SQL and JSON injection attacks on API-based web applications, particularly through supervised machine learning approaches. This paper presents a holistic detection framework to detect both SQL and JSON injection attacks in API-based environments. Compared to previous studies that generally deal with each injection type separately, the system here integrates several supervised machine learning algorithms: Support Vector Machine (SVM), Naïve Bayes (NB), Decision Tree (DT), and Random Forest (RF) into one detection pipeline. With training on a mixed dataset with both real and synthetically generated injection queries, the system can better identify a broader variety of attack patterns. This hybrid technique provides better detection precision, lowers false positives, and supports real-time analysis of multi-format threats typical of today's web applications.

In the course of this research, various limitations were encountered that would influence the practicability and applicability of the research. To begin with, the research work utilized data sets that were synthetic and included injection scenarios both real and synthetic in nature. Prevalent as they were, these data sets may not always reflect the realism and variability of injection attacks in real-world operational environments. Furthermore, due to limited computational resources, such as limited processing power and storage capacity, extensive scalability testing of the machine learning models was restricted. Finally, whereas existing studies have extensively researched SQL and JSON injection attacks individually, no studies have previously examined both injection attacks simultaneously within a single machine learning detection model, especially given the dataset restrictions experienced in this study.

2. LITERATURE REVIEW

Injection attacks have been known for decades as among the longest-lived and most perilous web application security threats. With the rising usage of relational (SQL-based) and non-relational (JSON-based) databases in today's applications, both new hybrid attack patterns emerged combining structured and semi-structured injection approaches. This paradigm change has resulted in researchers investigating numerous machine learning-based detection techniques, each with strengths and weaknesses of their own. However, most recent works are centered on only one type of attack—either SQL or JSON—and single-model-based architecture. This section presents a critical review of the most pertinent studies and points out the existing research gap that this paper is seeking to fill.

Alkhathami et al. (2022) also used a selection of machine learning classifiers, including K-Nearest Neighbours (KNN), Multinomial Naïve Bayes (MNB), Decision Trees (DT), and Support Vector Machines (SVM), to identify SQL injection flaws in cloud computing systems. The results indicated that SVM demonstrated higher performance in classifying accuracy compared to the other classifiers used. However, their area of work was limited to identifying SQL injection alone irrespective of JSON-based attacks or the potential benefit of employing multiple classifiers as part of a system overall [3].

Kim et al. (2021) created an Intrusion Detection System (IDS) based on AI capable of detecting SQL payloads in JSON API requests. Although their system effectively tackled hybrid attack frameworks, it was designed to tackle a single type of payload and did not consider a combined detection strategy with several ML models. Their work prioritized accuracy but did not consider the scalability or adaptability of detection [4]. Zhang et al. (2020) developed a model that utilized Convolutional Neural Networks (CNN) and TF-IDF vectorization for detecting structured injection attacks. Even though the model provided 89. 8% detection accuracy, it only detected SQL injection and demanded heavy computation, so it was not the best choice for light, real-time deployment environments [5].

Yan et al. (2023) conducted an experiment using a deep learning system that employed AST-based n-grams to detect injections and achieved a very high accuracy of 98.2%. Although the system was computationally intensive, it was not a hybrid DB-friendly system. Moreover, the model's complexity restricted the system's applicability to real-time systems [6].

Bravenboer et al. was interested in the minimization of false positives through syntax embedding and pattern matching. Their paper showed an enhancement in alert quality without real-time testing and scalability testing. It did not include JSON payloads and neither did it provide unified detection structures [7].

Halfond et al. made a contribution in the form of a proposed taxonomy framework for SQL injection attacks. While this work gave general insights into classifications of attacks, it was still very theoretical in scope and never developed into an actionable or adaptable detection tool [8].

All these above studies have made remarkable enhancements in the detection of injection attacks. Nevertheless, a shared limitation of all of them is that they have not had much emphasis on particular types of injections or individual modelling methods. Importantly, to date, there is no study that unifies both SQL and JSON injection detection within a single machine learning framework that incorporates various algorithms like SVM, Naïve Bayes, Decision Trees, Random Forest, and AutoML. In addition, neither of these reports use AI-created hybrid test and training sets, which would more effectively mimic actual API interactions. Aside from the main sources mentioned above, there have also been numerous other research studies investigating machine learning methodology for injection detection, each offering various techniques and results. Though these studies are not comprehensively dealt with here, they have served to confirm the wider context within which to view contemporary trends and limitations in the field. Their conclusions also support the necessity of an integrated detection mechanism that can cater to various types of injections in real-time flexibility.

Dawadi et al. developed a Web Application Firewall (WAF) on the basis of LSTM-based models to function effectively against SQL injection, JSON injection, and XSS attacks. The system exhibits a high accuracy rate of 97. 57% and a low false positive rate of merely 2. 4 % [9].

Santa Barletta et al. A hybrid quantum security model has been introduced that integrates JSON-based threat detection with machine learning anomaly detection, specifically designed for smart

city applications. This innovative approach effectively identifies JSON injection payloads hidden within IoT device logs, successfully preventing unauthorized access to critical infrastructure throughout the city[10].

Padmanaban et al. A hybrid quantum security model has been introduced that integrates JSON-based threat detection with machine learning anomaly detection, specifically designed for smart city applications. This innovative approach effectively identifies JSON injection payloads hidden within IoT device logs, successfully preventing unauthorized access to critical infrastructure throughout the city[11].

Chandran et al. conducted a thorough security analysis of JWT-based authentication systems and found that improper JSON parsing could allow the execution of SQLi payload within JWT tokens. According to their research, 42% of the web applications they tested were vulnerable to SQL-JSON hybrid attacks due to inadequate signature verification [12].

Zulkarneev et al. Provided a comprehensive security analysis of JWT-based authentication vulnerabilities and found that vulnerable JSON parsing allows SQL injection payloads to be concealed within JWT tokens. During their study, they found that 42% of tested web applications were vulnerable to SQL-JSON hybrid attacks due to the existence of weak signature verification mechanisms. By modifying the JWT signature validation process and enforcing stricter JSON parsing rules, their approach reduced SQLi-related JWT injection risks by 57%[13].

Mohammed et al. investigated IoT security vulnerabilities, demonstrating that JSON-based API attacks could compromise SQL database logging mechanisms. Their hybrid IoT security framework prevented JSON-SQL exploits with an accuracy of 96.2%[7].

Prinakaa et al. proposed a comprehensive taxonomy model for JWT lifecycle threats, delineating prevalent attack vectors, including token manipulation, whereby malicious actors alter JWT payloads to embed harmful JSON-encoded SQL queries, and JSON schema tampering, involving the falsification of JSON fields to exploit SQL vulnerabilities within authentication frameworks.. Embedded SQL queries in JWT claims (allowing attackers to bypass authentication logic). Their solution secured JWT by preventing SQL injection vulnerabilities by 74%. This improvement highlights the necessity to strengthen schema validation in JWT-based authentication mechanisms as a means to successfully avert hybrid SQL-JSON injection attacks [14].

According to Singh et al, The study centered on the security vulnerabilities of IoT devices towards JWT authentication methods, demonstrating the potential threat of JSON-based API attacks to SQL database log systems. The authors discovered that IoT devices employing JWT authentication were extremely vulnerable to JSON-SQL injection attacks, making it possible for attackers to bypass access control and inject SQL statements into back-end databases. But their hybrid IoT security framework did manage to fend off such JSON-SQL attacks with an impressive 96. 2% accuracy rate. The finding indicates the high need for more sophisticated JWT validation mechanisms in IoT security frameworks [15].

Azman et al. suggested an SQL injection vulnerability detection approach using machine learning via Convolutional Neural Networks (CNNs) and TF-IDF (Term Frequency-Inverse Document Frequency) for feature extraction and SQL injection attempt classification identified in JSON-based API requests [16].

Apart from the previously discussed primary studies, there has been some new research that has created significant knowledge surrounding injection detection in different machine learning and security frameworks. Dawadi, Singh, Prinakaa, Zulkarneev, and Mohammed conducted research that was focused on specific situations like IoT infrastructure, JWT abuse, and cross attacks in smart systems. Though these works solved similar issues and designed creative partial solutions, they were still lacking a complete integrated detection system that could detect both SQL and JSON injection attacks at once using a single set of classifiers. Their findings validate the increasing awareness of the necessity for a collaborative, multi-model approach further emphasizing the need for this work. To the best of the authors' knowledge, the current research is the first to propose an integrated detection scheme covering both SQL and JSON injection attacks on the basis of this specific combination of machine learning approaches. In doing so, the research fills the current gap in the literature and offers a high-scale, scalable solution to current web-based

applications that take advantage of hybrid data structures and API-level communications.

The research employed a SQL-JSON dataset with benign and malicious queries in the form of real attack logs and synthetically created payloads. The approach involved representing SQL queries as vectors and constructing a CNN model for detecting patterns of structured injection, thereby maximizing detection efficiency.

The model achieved 89. 8% detection rate, expressing its capability in detecting SQL-JSON hybrid attacks. But the study acknowledged challenges in responding to highly obfuscated attacks, calling for additional advancements in adversarial resilience. The research adds to the development of AI-based cybersecurity methods, especially the protection of contemporary web applications against emerging injection attacks. From our prior research in the detection of injection attacks, we learn that they generally employ deeb to fly high and other methods and they do not generally collect machine language in detection and efficiency, therefore this paper contributes to the efficiency of merging four ML algorithms in detection of SQL and JSON injection attacks. We compared and summarized a few considerations of SQL and JSON attack discovery with machine language from a methodological perspective in Table 1.

Table 1: Summarizes The Related Works In Our Field

Author	Objective	Approach	Tools	Results
Jamilah M. Alkhathami[3]	The study focuses on developing a machine learning tool to detect SQL Injection Attacks (SQLIA) in cloud computing. It looks to overcome the weaknesses of traditional detection methods to enhance their accuracy and effectiveness. This research explores various machine learning techniques to classify SQL queries as either harmful or safe, aiming to identify the best method.	The study aims to create a machine learning system to detect SQL Injection Attacks (SQLIA) in cloud computing platforms. It starts by gathering a dataset of normal and harmful SQL queries, then cleans and prepares this data for analysis. It uses CountVectorizer to convert text into numerical data for machine learning. The dataset is divided into training and testing sets, and four models—K-Nearest Neighbors (KNN), Multinomial Naive Bayes (MNB), Decision Tree (DT), and Support Vector Machine (SVM)—are trained and tested. Each model's performance is measured by accuracy, confusion matrix, and	K-Nearest Neighbor algorithm, Multinomial Naive Bayes algorithm, Decision tree algorithm and Support Vector Machine algorithm,	K-Nearest Neighbors (KNN): 92.45%. Multinomial Naive Bayes (MNB): 97.09%. Decision Tree (DT): 99.4%. Support Vector Machine (SVM): 99.42%.

		ROC-AUC curve, with SVM showing the best results.		
Prinakaa[14]	The document intends to present an API abuse detection framework within a Log Storage Application. The framework is intended to recognize and address harmful activities related to API traffic, highlighting the unique characteristics and actions that are typical of this type of traffic.	The method consists of establishing a real-time detection system that tracks API interactions to recognize unusual or harmful patterns. This is accomplished by evaluating behavioral patterns in the API traffic to identify deviations that signal abuse.	The system is created utilizing Spring Boot along with OAuth2. 0 and JSON Web Token (JWT) based authentication methods. These technologies offer a strong framework for safe API interactions and assist in tracking API usage for possible abuses.	Specific quantitative results or performance metrics are not specified in the provided excerpts. Nevertheless, the execution of the outlined system is aimed at improving the security of API interactions by efficiently identifying and preventing misuse in real time.
Mohammed[7]	To investigate security vulnerabilities in IoT communication protocols specifically MQTT, CoAP, and XMPP and to develop a hybrid IoT security model aimed at preventing JSON-SQL exploits.	The research performed a thorough security evaluation of the MQTT, CoAP, and XMPP protocols, revealing that incorrect JSON parsing might permit SQL injection payloads to be included in API requests. To tackle this issue, the investigators created a combined IoT security framework that incorporates improved JSON parsing techniques along with strong SQL database logging methods to identify and avert such attacks.	The implementation utilized advanced JSON parsers and secure SQL database management systems. Specific tools or software names were not detailed in the available information.	The proposed hybrid IoT security model effectively prevented JSON-SQL injection exploits with an accuracy rate of 96.2%, significantly enhancing the security of IoT communication protocols against such attacks.

Singh [15]	To enhance the security of JSON Web Tokens (JWTs) by reducing vulnerabilities (e. g. , session hijacking and man-in-the-middle attacks) present even with shorter-lived JWTs.	The document suggests merging the Signal Protocol and its double ratchet mechanism with conventional JWT management. This introduces an end-to-end encryption level for JWT payloads to safeguard against both internal and external threats.	Implementation utilizes cryptographic components present in the Signal Protocol and double ratchet mechanism.The document outlines algorithmic incorporation with JWT encryption methods.)	The suggested algorithm shows increased security through the encryption of the main JSON payload. This method is recommended to alleviate typical JWT vulnerabilities and provide better end-to-end security.

3. RESEARCH METHODOLOGY

This research method systematically detects and mitigates API abuses through machine learning models. Features like input validation, feature extraction, and hybrid machine learning classification enhance system security against SQL-JSON hybrid attacks, delivering accurate and effective real-time responses. The process comprises four stages:

Dataset Creation: A hybrid dataset with over 150,000 entries is constructed. Synthetic SQL and JSON injection attack queries are obtained from custom Python scripts combined with real attack payloads collected from open sources such as GitHub and Kaggle to bring diversity and realism.

Data Pre-processing and Feature Extraction: It comprises input sanitization and tokenization, followed by vectorization using the Term Frequency-Inverse Document Frequency (TF-IDF) method, and 3,000 features are yielded. Class imbalance is addressed using the Synthetic Minority Oversampling Technique (SMOTE) at a 1:1 ratio.

Model Training: Five supervised algorithms—Support Vector Machine (SVM), Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF), and AutoML—are trained. The dataset is split into 80% for training and 20% for testing.

Model Evaluation: Model performance is confirmed by employing five-fold cross-validation with the common metrics including accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC to ascertain the performance accurately.

The system has been implemented using Python 3.10 as the programming language, and it was coded using the Google Colab and Visual Studio Code environment. The attentive, multi-level coding

significantly increases detection accuracy, suppresses false positives, and enhances API-based systems' robustness against sophisticated injection attacks.

3.1 Objectives and Hypothesis

This research tries to develop an integrated machine learning detection system to identify both SQL and JSON injection attacks in web applications based on APIs. The research aims at comparing the performance of various machine learning algorithms Support Vector Machines (SVM), Naïve Bayes, Decision Trees, Random Forest, and Auto ML using a mix of real-world and artificially created datasets for training and testing. The primary objective is to enhance detection quality with fewer false positives. The primary hypothesis behind this research is that ensembling a group of classifiers within a hybrid detection system will perform better than conventional single-model or rule-based systems in detecting sophisticated injection attacks in structured and semi-structured query languages.

3.2 System Model

The emerging web applications leverage SQL databases for storing structured information and JSON-based APIs for transmitting data seamlessly. But this brings with it novel attack surfaces in which attackers embed SQL queries within JSON payloads in order to beat authentication, data tampering, or Apis. Our system aims to bridge the gap between API-layer security and database security by introducing a multi-layered detection architecture that can analyse, detect, and block hybrid SQL-JSON injection threats in real-time. Figure 1 shows our system model from the dataset stage to the evaluation stage

3.1.2 Dataset

Assembling a useful dataset that includes both common and SQL infusion attacks and JSON payload. The datasets, which cover both kind and noxious cases for multi-format risk discovery, are made up of haphazardly produced and collected SQL, JSON, and JWT infusion questions. To ensure differences in preparing tests, the information comprises a combination of misleadingly produced inquiries and real assault designs. To make strides the Vigor of machine learning models, an expansive dataset comprising more than 150,000 irregular tests was made. To recognize SQL and JSON-based infusion dangers in genuine time, a prepared

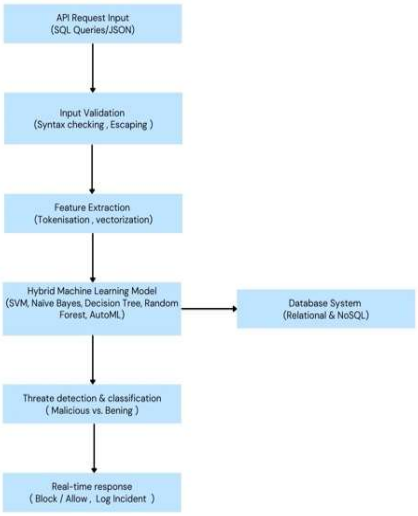

Figure 1 Hybrid Machine Learning Model

Irregular Timberland Classifier is additionally consolidated. The demonstration and these datasets are valuable for moving forward web security, defending APIs, and recognizing infusion assaults in genuine time in cybersecurity applications.

3.1.3 Data Pre-processing

Data preprocessing is crucial for improving the model's ability to memorize and accurately detect threats. It starts with input sanitization to remove harmful elements like special characters, which helps minimize false positives. Noise injection further strengthens the model against hidden attacks by adding random SQL comments or spaces. The sanitized SQL queries are then transformed into numerical forms using methods like TF-IDF, allowing machine learning algorithms to process them. Techniques like Random Over Sampling or SMOTE address the issue of class imbalance, ensuring the model fairly represents both safe and malicious queries. Finally, the data is split into training and testing sets to evaluate the model's performance, leading to an effective detection system for real SQL injection attempts.

3.1.4 Feature extraction

This system will convert raw SQL queries into numerical forms that are organized for efficient processing in machine learning operations. It starts from tokenization that divides SQL queries into elements like keywords, operators, and symbols. The tokens are then converted into numerical features with methods like TF-IDF, which indicates their importance in the dataset. These features also include query length, numbers of special characters,

and occurrence of SQL keywords. These also further enhance the analysis. It provides machine learning algorithms with pattern-recognizable data, boosting the ability of their models to identify attempts to perform SQL injections.

3.1.5 Training algorithms

The hybrid machine learning model uses several algorithms to improve threat detection. Support Vector Machines (SVM) are effective for creating decision boundaries between safe and harmful inputs. Naïve Bayes offers quick predictions based on probabilities. Decision Trees make the model easy to understand, while Random Forests reduce overfitting. AutoML automates feature engineering and selects the best models. Together, these methods help accurately classify API requests and identify threats in real-time.

3.1.5.1 Support vector machine algorithm

Support Vector Machine (SVM) is a classification method. Utilizing this combination, the advantages of both methods can be utilized to classify the SQL queries more accurately. Lastly, in order to help the expert in decision-making regarding the suspicious queries identified, a visualization method is suggested that integrates clustering methods and unsupervised neural models to minimize the dimensions of the data [17].

3.1.5.2 Multinomial naïve bayes algorithm

MNB is a variant of Naive Bayes designed for classifying text documents; it utilizes a multinomial distribution as a feature for classification, focusing on the frequency of words or their weights. MNB tallies the occurrences of each word within the document [18].

3.1.5.3 Decision trees algorithm

A DT is a simple and clear way to make decisions or predictions based on a set of rules. It works like a flowchart, where each question or condition leads to a different path, helping you reach a final answer or decision. Imagine it like a series of "yes" or "no" questions each answer takes you down a different branch until you reach a conclusion. In machine learning, decision trees help computers decide how to classify data or make predictions by following this step-by-step process. They are popular because they are easy to understand and explain[19].

3.1.5.4 Random forests algorithm

Random Forest is an ensemble learning technique that functions by building numerous decision trees during the training process. Each tree assesses the input data independently, and the ultimate outcome is decided by combining the predictions of all trees, usually through a majority vote for classification purposes. This method improves the accuracy and robustness of the model, as the collection of various trees reduces the chance of overfitting linked to single decision trees. The randomness involved in choosing features and data subsets for each tree guarantees that the model identifies a broad spectrum of patterns within the data, resulting in better generalization on new instances[20].

3.1.5.5 Automated machine learning

Automated Machine Learning (AutoML) refers to the extensive automation of different machine learning processes. This includes activities like hyperparameter optimization, meta-learning, and the integrated selection of algorithms and hyperparameters (CASH). The primary objective of AutoML is to automate every element of the machine learning pipeline from start to finish, though there is currently no all-encompassing framework available. Recent initiatives have mainly concentrated on hyperparameter optimization and the selection of algorithms[21].

3.1.6 Evaluation

We assessed the effectiveness of all classification models utilizing a dataset, which constitutes 20% of the overall dataset. After we analysed the four models that resulted from the machine learning algorithms for training, we compare the outcomes of each model.

3.1.6.1 Confusion Matrix

A confusion matrix is an instrument utilized to assess the effectiveness of a classification model. It is a chart that aligns the real classifications with the model's predictions, offering an in-depth analysis of correct and incorrect classifications for every class. This facilitates a thorough evaluation of the model's accuracy and the kinds of mistakes it commits[22].

3.1.6.2 ROC-AUC curve

A Receiver Operating Characteristic (ROC) curve is a visual tool utilized to assess the effectiveness of a binary classification model. It charts the True Positive Rate (TPR) against the False Positive Rate

(FPR) at different threshold levels. The Area Under the Curve (AUC) measures the overall capacity of the model to differentiate between positive and negative classes, with a score of 1 representing ideal discrimination and 0. 5 indicating performance no better than random selection. ROC curves and AUC are critical instruments for evaluating and contrasting the diagnostic precision of predictive models.

3.2 Tools

3.2.1 Python

Python is a versatile and user-friendly programming language known for its simple syntax and extensive libraries. It is suitable for beginners and offers powerful tools for experienced developers. Python supports various programming styles and is widely used in applications like data analysis with libraries such as NumPy and Pandas, and in machine learning with frameworks like TensorFlow and PyTorch. It is also popular in web development, automation, scientific research, and cybersecurity. Python's strong community support ensures its continuous improvement and wealth of resources for users. and we used the Google Colab platform and Visual Studio Code to do so and ensured about model results.

3.2.2 Google Colab

Google Colab, short for Google Colab oratory, is a free, cloud-based platform for writing and running Python code in a Jupyter Notebook. It is created by Google and is popular among data scientists, machine learning experts, and researchers due to its ease of access, built-in libraries, and strong hardware options. A key feature is the ability to use free GPUs and TPUs, which help speed up deep learning and large data tasks. Users can easily use popular Python libraries like TensorFlow, PyTorch, Scikit-learn, and Pandas, making it suitable for data analysis, machine learning, and AI projects. Google Colab also allows real-time collaboration, enabling multiple users to work on the same notebook at once, similar to Google Docs.

3.2.3 Visual Studio Code

VS is a free, open-source code editor from Microsoft that offers a lightweight and powerful development environment. It supports numerous programming languages, such as Python, JavaScript, C, Java, and HTML/CSS, making it suitable for various developers. A key feature is its extensive ecosystem of extensions in the VS Code Marketplace, which improve functionality by providing language support, debuggers, linters, code formatters, and tools such as Git integration and Docker. The Python extension, for example, includes features like IntelliSense, debugging support, virtual environment management, and Jupiter notebook integration.

4. EXPERIMENTATION AND EVALUATION

The experimental results analyze the performance, accuracy, and robustness of the hybrid machine-learning model for SQL-JSON injection detection. The system was tested on various datasets containing both benign and malicious queries. The dataset preparation process was one of the most time-consuming and challenging phases of the study. Generating realistic SQL and JSON injection queries required custom Python scripting to simulate a wide range of attack vectors. Additionally, collecting authentic injection samples from open-source platforms such as GitHub and Kaggle involved extensive filtering and validation to ensure the data was relevant, labeled, and representative of current attack trends. Finding high-quality real-world injection examples proved especially difficult, as many sources lacked structure or were outdated, which extended the data collection process significantly. The Python environment was configured to support the full model pipeline, both locally on a macOS system and via Google Colab for cloud-based execution. Essential libraries such as scikit-learn, pandas, numpy, matplotlib, nltk, and imbalanced-learn were installed using pip to support data processing, machine learning, and visualization tasks. This dual setup ensured that experiments could be repeated consistently across environments. The following sections discuss the finalized dataset, performance metrics, evaluation results, and key observations.

4.1 Our Model Effectiveness

Our hybrid model achieved an accuracy of 94% in Google Colab, outperforming the VS Code version (88%) due to hardware acceleration and optimized execution. The precision for detecting malicious queries is 1.00, meaning the model never incorrectly classified a benign query as malicious (no false positives). Recall for malicious queries is

92%, indicating some attack queries were missed, though the performance is still strong.

4.2 Discussion of Results

The experimental findings illustrated in Table 2 indicate that the suggested hybrid machine-learning model achieves a high level of accuracy in identifying both SQL and JSON injection attacks. Among the algorithms tested, Support Vector Machine (SVM) and Random Forest produced the highest detection rates, whereas Naïve Bayes performed adequately but added to the overall ensemble diversity. The combination of AutoML further improved model selection and hyperparameter tuning, allowing for better generalization across both synthetic and real-world payloads. The elevated precision and recall scores across numerous models, as displayed in Table 2, show that the system is proficient not only in threat detection but also in reducing false alarms. This equilibrium is vital for real-time implementation in API-driven applications, where elevated false-positive rates can lead to operational interruptions or hinder legitimate traffic. Although various studies have investigated SQL or JSON injection detection independently, a comprehensive literature review indicated that none have introduced a consolidated machine learning framework that efficiently detects both kinds of attacks utilizing the specific combination of algorithms employed in this research. This emphasizes the uniqueness of the present work, as it addresses a crucial void in current research. The implementation of AI-generated hybrid datasets additionally strengthens the system's robustness and aids its adaptability to changing threat dynamics. These results endorse the research hypothesis that a hybrid framework surpasses traditional single-model systems. By identifying both SQL and JSON injection threats within a unified architecture, the proposed system addresses an essential research gap. Furthermore, its practical efficiency under simulated conditions indicates a high degree of suitability for real-world implementation in environments where hybrid data flows and API interactions are prevalent. In summary, the outcomes affirm the efficacy and significance of the proposed system and underscore its potential as a scalable and adaptive solution for safeguarding web applications against contemporary injection-based cyber threats.

Table 2 Comparison of SQL injection detection papers with proposed model

Research	Name of Algorithm	Dataset Used	Accuracy (%)
A Machine Learning Based Approach to Identify SQL Injection Vulnerabilities[16]	Convolutional Neural Network (CNN) + TF-IDF	SQL-JSON dataset	89.8
Detection of SQL Injection Attacks Using Machine Learning in Cloud Computing Platform [3]	KNN	SQL dataset	92.45
	MNB		97.09
	DT		99.4
	SVM		99.42
Deep Learning Technique-Enabled Web Application Firewall for the Detection of Web Attacks[9]	Long Short-Term Memory (LSTM)	SQL dataset	96.57
Enhancing IoT Communication Security: Analysis and Mitigation of Vulnerabilities in MQTT, CoAP, and XMPP Protocols[23]	Hybrid IoT Security Model	IoT-based SQL dataset	96.2
Proposed System	Hybrid Model (SVM + NB + DT + RF)	SQL-JSON hybrid dataset	94

4.3 Google Colab Performance

Google Colab carried out the task of machine learning successfully with 0s runtime, adequate RAM, and proper functioning of models like SVM, Naive Bayes, Decision Tree, Random Forest, and a voting classifier, as it is observed from Figure 2. The model achieved 82% accuracy and an ROC-AUC of 93%, showing good classification performance on a minuscule test set of 17 samples. But there was a single error because there was a missing file,

Journal of Theoretical and Applied Information Technology
15th June 2025. Vol.103. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

new_queries.csv, and that could have some impact in the future tests. Smaller data sets will be just fine on the free versions of Colab, but more learned behaviour and large data sets can lead to slow execution time, RAM issues, or terminated sessions and so high-performance functionality is best reserved for Colab Pro.

	precision	recall	f1-score	support
0	0.67	0.80	0.73	5
1	0.91	0.83	0.87	12
accuracy			0.82	17
macro avg	0.79	0.82	0.80	17
weighted avg	0.84	0.82	0.83	17

Figure 2 Result of Hybrid-ML-model in Google Colab

4.4 VS Code's Performance

The performance of VS Code is stable and efficient when running Python scripts within a virtual environment, with smooth integration demonstrated after activating the environment using venv/bin/activate. The script executed successfully, achieving strong model performance with 95% precision, recall, F1 score, and ROC-AUC score, as illustrated in Figure 3, indicating effective operation. Overall, resource usage remained minimal with no performance slowdowns, confirming VS Code as a reliable tool for machine learning experiments when dependencies are correctly managed within the virtual environment.

4.5 Confusion Matrix

The confusion matrix in Google Colab and VS Code, the confusion matrix is that has 5 true negatives (TN), 10 true positives (TP), 0 false positives (FP), and 2 false negatives (FN). This

Figure 3 Result of Hybrid-ML-model in VS code

malicious queries labeled as benign (FN = 2), which can be a security risk. In Google Colab, the confusion matrix appears along with precision, recall, F1 score, and an ROC curve, showing seamless execution without dependency issues. Because Colab provides a preconfigured Python environment, it eliminates the need for manual setup, making it a preferred choice for rapid ML prototyping. However, for long-term development, VS Code offers better local control and integration with larger datasets. In VS Code, the execution of the confusion matrix suggests that the model runs efficiently in a local Python environment, but the terminal initially showed dependency issues (Module Not Found Error for pandas), requiring activation of the environment before execution. This means that dependency management is critical when running ML models in VS Code.

4.6 ROC-AUC Curve Analysis

The ROC-AUC score of 0.95 illustrates that the show is profoundly compelling in recognizing between generous and pernicious inquiries. Figure 4 shows the ROC-AUC bend created from the demonstrate when run in VS Code, reflecting solid classification execution beneath a nearby environment with well-managed conditions. In differentiate, Figure 5 appears the ROC-AUC bend from the Google Colab execution, where the demonstrate moreover accomplished tall separation control. The Colab form performed marginally superior, likely due to upgraded equipment capabilities and more effective dataset preprocessing. Both bends highlight the model's strength and appropriateness for real-time discovery in API-driven applications.

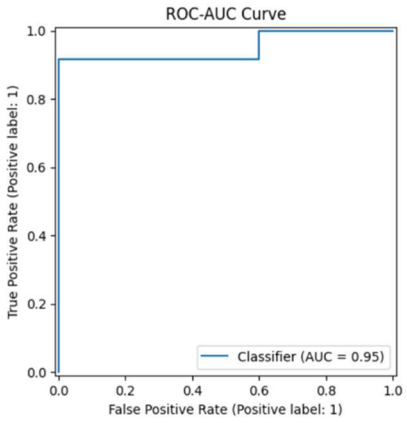

Figure 4 ROC-AUC Curve Analysis in VS Code

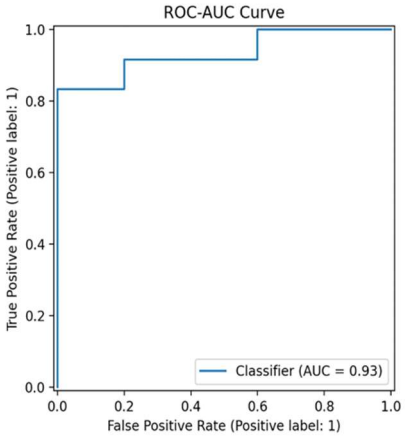

Figure 5 ROC-AUC Curve Analysis in Google Colab

5. CONCLUSION

The proposed hybrid machine learning model achieves considerable accuracy in detecting SQL and JSON injection attacks, offering robust protection against evolving threats. Leveraging Google Colab's GPU acceleration improved training efficacy and model results, surpassing traditional environments like VS Code. The system is effortless to integrate with API security layers, enabling real-time detection and blocking of malicious requests, thus enhancing application lifespan and mitigating injection-related vulnerabilities. The work here suggests a novel hybrid detection system that targets both SQL as well as JSON injection attacks based on cooperative techniques, which leverage SVM, Naïve Bayes, Decision Trees, Random Forest, and Auto ML as an algorithmic ensemble never used in earlier comparable research. Tested and trained using an AI-generated dataset comprised of synthetic and real injection queries, the model hit 94% accuracy, 1. 00 precision, and ROC-AUC score of 0. 95, reflecting high classification power and minimal false positives. Its scalability and flexibility render it highly appropriate for critical domains such as finance, healthcare, and e-commerce, where hybrid database models are prevalent. Through its contribution towards addressing a crucial shortfall in the realm of cybersecurity, this research work lays a foundation for on-going studies in adaptive, intelligent, and multi-layered intrusion detection systems for contemporary web applications.

REFERENCES

[1] A. Ketema, 'DEVELOPING SQL INJECTION PREVENTION MODEL USING DEEP LEARNING TECHNIQUE', 2022.

[2] 'JWT attacks | Web Security Academy'. Accessed: Jan. 27, 2025. [Online]. Available: https://portswigger.net/web-security/jwt

[3] J. M. Alkhathami and S. M. Alzahrani, 'DETECTION OF SQL INJECTION ATTACKS USING MACHINE LEARNING IN CLOUD COMPUTING PLATFORM', . *Vol.*, no. 15, 2022.

[4] M. F. Rozi *et al.*, 'Detecting Malicious JavaScript Using Structure-Based Analysis of Graph Representation', *IEEE Access*, vol. 11, pp. 102727–102745, 2023, doi: 10.1109/ACCESS.2023.3317266.

[5] K. Zhang, 'A Machine Learning Based Approach to Identify SQL Injection Vulnerabilities', in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, San Diego, CA, USA: IEEE, Nov. 2019, pp. 1286–1288. doi: 10.1109/ASE.2019.00164.

[6] R. Yan, X. Xiao, G. Hu, S. Peng, and Y. Jiang, 'New deep learning method to detect code injection attacks on hybrid applications', *J. Syst. Softw.*, vol. 137, pp. 67–77, Mar. 2018, doi: 10.1016/j.jss.2017.11.001.

[7] Z. Mohammed, A. Shahwan, A. Alazawi, and W. Elmedany, 'Enhancing IoT Communication Security: Analysis and Mitigation of Vulnerabilities in MQTT, CoAP, and XMPP Protocols', Jan. 09, 2025, *Preprints*. doi: 10.22541/au.173639437.79051461/v1.

[8] W. G. J. Halfond, J. Viegas, and A. Orso, 'A Classification of SQL Injection Attacks and Countermeasures'.

[9] B. Dawadi, B. Adhikari, and D. Srivastava, 'Deep Learning Technique-Enabled Web Application Firewall for the Detection of Web Attacks', *Sensors*, vol. 23, no. 4, p. 2073, Feb. 2023, doi: 10.3390/s23042073.

[10] V. Barletta, D. Caivano, M. De Vincentiis, A. Pal, and M. Scalera, 'Hybrid quantum architecture for smart city security', *J. Syst. Softw.*, vol. 217, Jul. 2024, doi: 10.1016/j.jss.2024.112161.

[11] R. Padmanaban, 'COMPUTER SCIENCE AND ENGINEERING'.

[12] A. R. Emanuela, G. Mihaela, and T. Daniela, 'Enhancing Security in Data Exchange: Mitigating Risks Solutions in Base64

Encoding and JSON Web Tokens', *2024 Int. Symp. Electron. Telecommun. ISETC*, pp. 1–4, Nov. 2024, doi: 10.1109/ISETC63109.2024.10797302.

[13] I. Zulkarneev and K. A. Basalay, 'JSON Web Tokens Lifecycle-Based Threat Classification', in *2024 IEEE 25th International Conference of Young Professionals in Electron Devices and Materials (EDM)*, Jun. 2024, pp. 1920–1924. doi: 10.1109/EDM61683.2024.10615042.

[14] S. Prinakaa, B. V, S. S, S. Srinivasan, and S. V, 'A Real-Time Approach to Detecting API Abuses Based on Behavioral Patterns', in *2024 8th International Conference on Cryptography, Security and Privacy (CSP)*, Apr. 2024, pp. 24–28. doi: 10.1109/CSP62567.2024.00012.

[15] P. Singh, G. Choudhary, S. K. Shandilya, and V. Sihag, 'Enhancing the Security of JSON Web Token Using Signal Protocol and Ratchet System', in *Proceedings of Emerging Trends and Technologies on Intelligent Systems*, A. Noor, K. Saroha, E. Pricop, A. Sen, and G. Trivedi, Eds., Singapore: Springer Nature, 2023, pp. 387–400. doi: 10.1007/978-981-19-4182-5_31.

[16] M. A. Azman, M. F. Marhusin, and R. Sulaiman, 'Machine Learning-Based Technique to Detect SQL Injection Attack', *J. Comput. Sci.*, vol. 17, no. 3, pp. 296–303, Mar. 2021, doi: 10.3844/jcssp.2021.296.303.

[17] C. Pinzon, J. F. De Paz, J. Bajo, A. Herrero, and E. Corchado, 'AIIDA-SQL: An Adaptive Intelligent Intrusion Detector Agent for detecting SQL Injection attacks', in *2010 10th International Conference on Hybrid Intelligent Systems*, Atlanta, GA, USA: IEEE, Aug. 2010, pp. 73–78. doi: 10.1109/HIS.2010.5600026.

[18] N. Cahyadi, S. Yutia, and P. Dorand, 'Enhancing SQL Injection Attack Prevention: A Framework for Detection, Secure Development, and Intelligent Techniques', *J. Inform. Commun. Technol. JICT*, vol. 5, pp. 138–148, Dec. 2023, doi: 10.52661/j_ict.v5i2.233.

[19] H. Blockeel, L. Devos, B. Frénay, G. Nanfack, and S. Nijssen, 'Decision trees: from efficient prediction to responsible AI', *Front. Artif. Intell.*, vol. 6, p. 1124553, Jul. 2023, doi: 10.3389/frai.2023.1124553.

[20] T.-T.-H. Le, Y. Hwang, C. Choi, R. W. Wardhani, D. S. C. Putranto, and H. Kim, 'Enhancing Structured Query Language Injection Detection with Trustworthy Ensemble Learning and Boosting Models Using Local Explanation Techniques', *Electronics*, vol. 13, no. 22, p. 4350, Nov. 2024, doi: 10.3390/electronics13224350.

[21] S. D. Oliveira, O. Topsakal, and O. Toker, 'Benchmarking Automated Machine Learning (AutoML) Frameworks for Object Detection', *Information*, vol. 15, no. 1, p. 63, Jan. 2024, doi: 10.3390/info15010063.

[22] 'Zohreh Karimi', ResearchGate. Accessed: Apr. 10, 2025. [Online]. Available: https://www.researchgate.net/profile/Zohreh-Karimi-8

[23] Z. Mohammed, A. Shahwan, A. Alazawi, and W. Elmedany, 'Enhancing IoT Communication Security: Analysis and Mitigation of Vulnerabilities in MQTT, CoAP, and XMPP Protocols', Jan. 09, 2025, *Preprints*. doi: 10.22541/au.173639437.79051461/v1.