1. Introduction

As reliance on technology grows exponentially, the surface for cyber-attacks continues to expand. One prevalent threat plaguing the online landscape is SQL injection, which exploits vulnerabilities in database-driven applications. Attackers craft malicious SQL code to be passed through victim websites, potentially gaining access to sensitive databases (Muslihi, & Alghazzawi, 2020). This review comprehensively analyzes a thesis investigating machine learning as a solution for SQL injection detection on e-commerce platforms. The thesis aimed to systematically examine SQL injection vectors and methods, then evaluate selected machine learning algorithms against real and synthetic datasets representing such attacks.

E-commerce platforms, storing vast amounts of sensitive customer and financial data, have become major targets for cyber-attacks. SQL injection is a particularly prevalent and dangerous attack technique, exploiting vulnerabilities in dynamic SQL queries to manipulate databases and potentially compromise entire systems. While input validation and parameterized queries help defend against basic SQL injection, more sophisticated methods like blind and time-based variations can still bypass these protections. Machine learning shows promise as an advanced detection mechanism capable of adapting to the evolving tactics of skilled attackers.

Five key algorithms—Logistic Regression, Naive Bayes, Random Forest, Artificial Neural Network, and hybrid models—were empirically analyzed based on metrics like accuracy, precision, and recall. A significant finding was Random Forest's robust performance in balancing both precision and recall, highlighting its suitability for accurately identifying malicious queries, (Alghawazi et al., 2022). The algorithms were rigorously tested on real SQL queries obtained from hacker communities as well as synthetic data generated using domain knowledge of injection patterns and strategies. Preparing and executing comparative experiments of each model when trained on balanced and imbalanced datasets helped identify the advantages and drawbacks of these algorithms for the indicated threat area.

Despite there being no single top solution, Random Forest independently proved to be the most accurate at correctly representing the proportion of relevant cases while having the best average recall, demonstrating its usefulness in complicated scenarios such as multistage attacks. However, limitations such as reliance on a single dataset and the high computational requirements of some methods were identified.

This comprehensive review seeks to thoroughly analyze the thesis' methodology, results, and conclusions to provide cybersecurity practitioners with guidance on using machine learning for SQL injection detection. By examining the work's strengths and weaknesses, recommendations will be proposed to address gaps through future research efforts. The overarching aim is to improve defenses against this ongoing cyber threat.

2. Literature Review

2.1. Machine Learning for SQL Injection Detection: A Review of the Literature

2.1.1. Overview of SQL Injection and Machine Learning Approaches

In web application development, SQL injection is one of the toughest challenges because it involves server-side applications that communicate with database systems. Injection attacks exploit weaknesses in the application's handling of user-supplied inputs to execute unauthorized SQL statements (Meng et al., 2021). Although other mitigation techniques for SQL injection, such as input validation, exist, they have their own demerits and are often inadequate in handling more complex injection approaches. The feasibility of applying machine learning for real-time identification of injection attempts has been highlighted by various authors (Sahu et al., 2022).

In this section, we briefly discuss the background on SQL injection and provide an overview of machine learning-based detection, which will be discussed in subsequent sections. We cover the basics of SQL injection, describe popular injection vectors and methods, and review various approaches (Ruiz et al., 2019). Additionally, a brief introduction to the machine learning concept is provided, with a review of previous studies on using machine learning approaches for injection identification (Dhanalakshmi et al., 2021).

2.1.2. Common SQL Injection Vectors and Techniques

They are usually the points of injection attacks because they can interface with backend databases. Ruiz et al. (2019) categorized common vectors according to forms such as login, search, or feedback/comment forms. Through these entry

points, an attacker can simply attach malicious SQL statements to otherwise legitimate SQL calls to disrupt the delicate database cycle.

The techniques associated with injections have also become more refined. Earlier techniques such as error injection and union query injection still exist, but they have evolved into blind injection and time-delay injection methods, making them harder to detect. In error-based injections, the attacker manipulates the syntax of the database query with invalid statements to elicit an error response, while union queries inject credible queries to serve as a vessel for malicious payloads. Blind SQL injections often check query success by observing changes in the application's behavior, such as responses triggered by injected tautologies or variations in response times caused by time-delay techniques (Hussain et al., 2021).

2.1.3. Machine Learning Techniques for Classification

Logistic regression is stated to be one of the most popular algorithms in machine learning-based classifiers for classification issues (Halbouni et al., 2022). The logistic function is used to predict probabilities on binary classification datasets. One of the advantages of logistic regression is that its implementation is easy, and the interpretation of its findings is straightforward (Ruiz et al., 2019). However, it can struggle with very large or complex datasets as it is prone to overfitting (Dhanalakshmi et al., 2021). Additionally, feature engineering is required to transform raw data into parameters the model can understand.

Table 1 Comparison of machine learning algorithms for classification table format

Algorithm	Description	Strengths	Limitations
Logistic Regression	Estimates probabilities using a logistic function to model binary classification problems	Simple to implement and interpret; handles unbalanced classes well	Can overfit on complex problems; requires feature engineering
Random Forest	Ensemble of decision trees that vote on the most popular class	High accuracy; handles unbalanced classes and non-linear/complex problems well	Requires hyperparameter tuning; less interpretable than single models
Neural Networks	Loosely mimics biological neural structure with interconnected nodes	Automatically learns complex patterns; strong for natural language/unstructured data	Prone to overfitting; requires large labelled datasets; hardest to interpret

As shown in Table 1, algorithms like logistic regression, random forest, and neural networks have all demonstrated applicability to the problem. Random forest stands out for balancing interpretability with strong performance on complex, unbalanced problems like those seen in injection detection.

Random forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees (Hussain et al., 2021). Studies have shown that it achieves high accuracy on complex datasets even when some variables are irrelevant (Matheickal et al., 2021). The use of multiple decision trees helps reduce overfitting and makes the approach more robust to outliers and noise in the data compared to single models. A limitation is that some hyperparameter tuning may be required to achieve optimal performance.

Neural networks attempt to mimic the human brain by using interconnected nodes that work together to learn from large amounts of data (Meng et al. 2021). When applied to classification problems, they can automatically learn nonlinear and complex patterns from raw features without much data pre-processing. However, training neural networks requires substantial computational resources and large labelled datasets which may limit its practical use (Sahu et al. 2022). The models also tend to be complex black boxes making interpretability challenging.

Support vector machines (SVM) find a hypersurface in a multidimensional feature space that distinctly classifies data points (Ruiz et al. 2019). They work well for text classification and are effective even when the training dataset is small. However, SVMs do not directly provide probability estimates, which are often desired, and do not scale well with large datasets as training times increase rapidly with data size.

2.1.4. Evaluation of Machine Learning Models

Proper evaluation of machine learning models is important both during development and after deployment to understand how well a model performs in practice. A variety of metrics are commonly used to assess classification effectiveness (Sahu et al., 2022). Precision examines a model's ability to return only relevant instances by measuring the number of true positives against the total number of predicted positive instances (Ruiz et al. 2019). While precision shows how well a model avoided false positives, it does not account for instances that were incorrectly predicted as negative.

Recall, also known as sensitivity, specifically probes a model's ability to detect all positive instances within a dataset (Dhanalakshmi et al. 2021). It compares the number of true positives to the total number of actual positive instances, accounting for false negatives in addition to true positives. High recall indicates fewer positive instances were missed but does not guarantee precision, since some negatives may have been misclassified as well.

To satisfy both precision and recall, the F1 score computes the harmonic mean of the two offering an efficiency measurement for what it seeks to achieve, as noted by Hussain et al. (2021). When set at its best value of 1, this shows perfect accuracy of precision and recall in which every instance gets detected and correctly tagged. The F1 score takes into account both potential false positives and false negatives simultaneously, providing a more comprehensive assessment of performance.

Table 2 a Confusion Matrix

	Predicted Negative	**Predicted Positive**
Actual Negative	True Negative (TN)	False Positive (FP)
Actual Positive	False Negative (FN)	True Positive (TP)

Table 2 b Confusion Matrix for Logistic Regression Model with and without data balancing

	Imbalanced data		**Balanced data**	
	Positive	**Negative**	**Positive**	**Negative**
Positive	TP3900 (63.1%)	FP 13 (0.2%)	TP 3891 (49.8%)	FP23 (0.3%)
Negative	FN 45 (0.7%)	TN 2223 (36.0%)	FN 81 (0.1%)	TN 3817 (48.9%)

Table 2 (a) above shows the precision and recall using the depiction of the confusion matrix as illustrated in Table 2 (b) above (Matheickal et al., 2021). This 2x2 matrix of true positives, true negatives, false positives, and false negatives is useful for dissecting how the models separate the signal from the noise and serves as a blueprint for this work. From this perspective, as well as using accuracy together with other indicators such as precision, recall, and the F1 measure, models can be compared and tested based on their real classification capability.

2.2. Black Box ML methods for SQL Injection Detection

2.2.1. Support Vector Machines for SQL Injection Detection

Support Vector Machines (SVMs) are vital classification machine learning models that map data points into an 'n'-dimensional space through vectors and segregate classes using hyperplanes (Das et al., 2020). SVMs have been frequently used for classification problems like SQL injection detection. In research conducted by Jha et al., SVMs were used to determine whether a query is benign or malicious. The model was trained on a dataset of over 10,000 queries from the most famous websites and achieved an accuracy of above 97.5%. A distinct advantage of SVMs is their capacity to search for nonlinear relationships using kernel functions, making them well-suited for large data samples. However, a disadvantage of SVM models is that they do not offer probability estimates, which may be required in production models, as highlighted by Dash and Paul (2021).

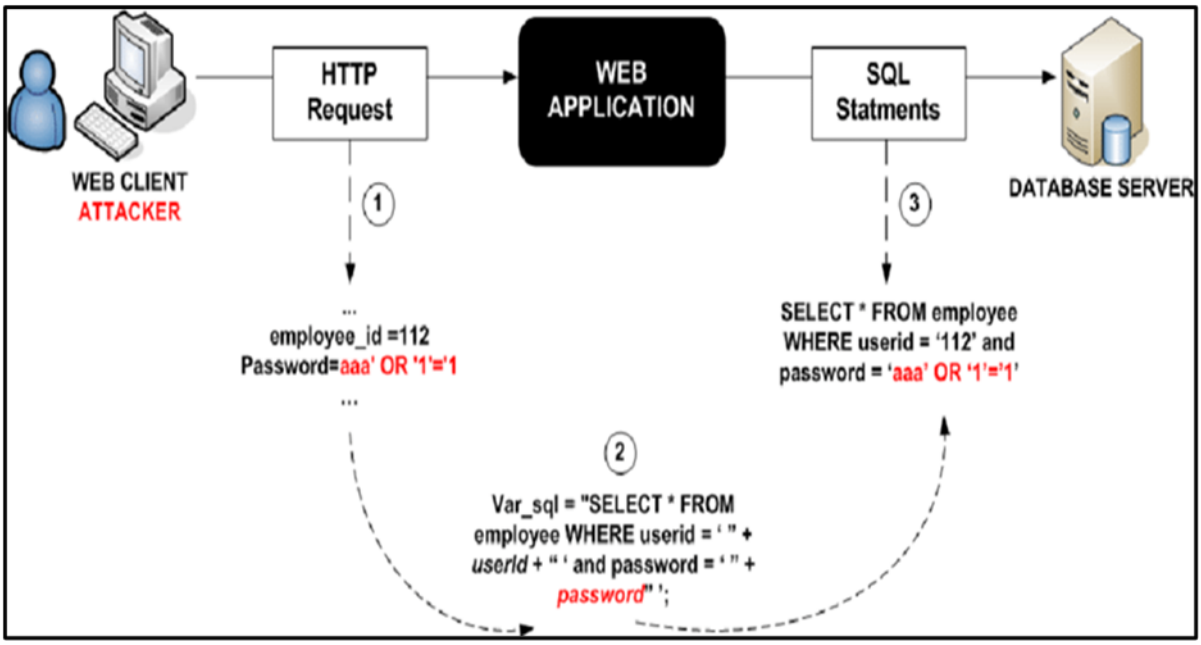

Figure 1 SQL injection Life Cycle (Purushottam & Pundlik, 2014)

Another area where SVMs have been used is ensemble modeling. It becomes clear that by integrating SVMs with other techniques, such as Decision Trees and Neural Networks, it is possible to further enhance the overall performance. Zhao et al. (2019) proposed stacking SVMs through several rounds using Gradient Boosting, with somewhat higher precision compared to a single SVM. These ensembles are based on the fact that while each SVM has its limitations, their combinations can mitigate these shortcomings. However, the creation of near-optimal ensemble models requires the careful tuning of many parameters, which can be time-consuming (Das et al., 2020).

SVMs are also well-suited for imbalanced classification problems, often seen in security domains like SQL injection detection, where malicious instances are fewer. Sreelekha and Aravindan (2018) applied sampling techniques to address class imbalance before training SVMs on query logs. This preprocessing step enhanced model metrics such as the F1 score and geometric mean, which are good measures for skewed data. Nevertheless, algorithm selection and configuration significantly impact performance on non-linearly separable imbalanced problems.

2.3. Sources of SQL Injection Attacks

User input forms a primary source of SQL injections, as attackers can craft malicious strings targeting vulnerable web forms (Shachi et al., 2022). Input fields like search bars, login pages and feedback mechanisms are routinely targeted (Kindy, & Pathan, 2011, June). Insufficient validation of these untrusted sources enables injections to manipulate backend queries (Uwagbole et al., 2017, May). Cookies also pose security risks when injected strings are passed to servers uninspected (Vishnu et al., 2022). Manipulating server-side variables such as HTTP headers exploits validation vulnerabilities by altering SQL commands unnoticeably (Erdődi et al., 2021).

Usage of outdated software with known vulnerabilities invites attacks, as patches often resolve flaws (Purushottam & Pundlik, 2014). Regular updates addressing weaknesses are necessary to close avenues of exploitation (Purushottam & Pundlik, 2014). Secondary interactions like file uploads can inject SQL through interfaces beyond the main application (Srivastava et al., 2023). Automated scanning exposes innumerable Internet-facing services to vulnerability identification, allowing opportunistic assaults requiring minimal technical skill (Srivastava et al., 2023). Mitigations must account for both immediate and ancillary SQL entry points.

2.4. Impact of SQL Injection Attacks

SQL injections often result in the exposure of sensitive customer and organizational records. Data breaches that release identities, finances, and private communications damage individual privacy (Zhu, et a., 2023 October). The 2017 Equifax leak of 147 million users' private and financial data demonstrates the far-reaching implications of coding vulnerabilities (Alkhathami & Alzahrani, 2022). Organizations experience reputational effects through diminished customer loyalty

and stock depreciation post-breach (Zhang, 2019). Monetary costs also burden victim companies through litigation, compliance, forensic auditing, customer turnover, and lost business (Alkhathami & Alzahrani, 2022).

Manipulation of databases enables adversaries to inflict financial harm by modifying transaction histories and account balances (Falor et al., 2022). Disruption to key services denies availability by overloading database resources with intensive queries (Ramasubramanian, & Kannan, 2006). Complete server takeover grants control to delete sensitive records or introduce malware, infecting connected infrastructure (Abdulhamza, & Al-Janab, 2022). Service downtime strains operational continuity and incurs recovery fees (Alkhathami & Alzahrani, 2022). Losses significantly impact smaller businesses without robust security budgets. Overall, SQL attacks threaten the confidentiality, integrity, and accessibility of critical networked data.

2.5. Web Application Architecture and Security

The architecture of web applications directly affects their security and vulnerability to SQL injections (KRISHNA, & Gopinath, 2022). A multilayer architecture segregates components into discrete user interface, application logic, and data tiers for optimal performance (Uwagbole et al., 2017). The front end, utilizing HTML, CSS, and JavaScript, delivers content, while a backend tier implements domain logic independently using Object Relational Mapping (ORM) or stored procedures (Tang et al., 2020). The main principle of separation is that it takes the logic layer out of the user layer so that it cannot be affected by any manipulation (Luo et al., 2019).

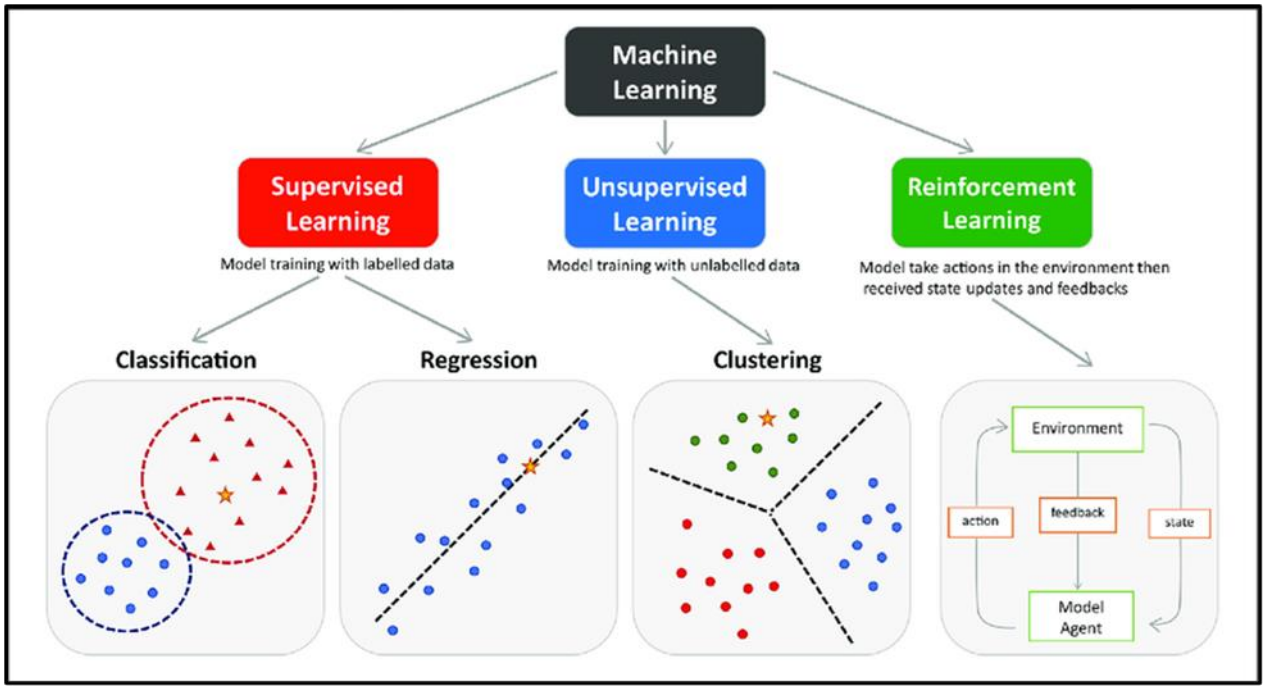

Figure 2 Machine Learning Algorithm

Database conversion is accomplished by an abstraction layer and does not directly use SQL queries (Luo et al., 2019). ORM abstracts the interactions by providing a higher-level object persistence interface in the form of object relational mapping, thus minimizing the direct use of database operations (Uwagbole et al., 2017). Stored procedures help by allowing parameterized, strongly typed queries, which prevent SQL commands from being written in the application code (Tang et al., 2020). Whitelist techniques ensure that input validation only allows characters that have been whitelisted at the entry points (Tang et al., 2020). Escaping enables output encoding, ensuring special symbols displayed do not result in broken markup execution (Srivastava et al., 2023). Proper integration of defensive tactics, such as the principles of least privilege and fail-safe defaults, enhances security.

Sharpening user identity and attribute access controls involves determining user identities and their subsequent access rights through proper authentication, authorization, and session management (Srivastava et al., 2023). Effective use of web-based applications should be modular, loosely coupled, involve information hiding, and maintain low trust boundaries to build up resilience against failures while remaining highly resistant to attacks (Purushottam & Pundlik, 2014). Future-proofing requires continual review and updating as threats evolve over time (Kamtuo & Soomlek, 2017).

A holistic strategic process addresses the root causes and consequences of vulnerabilities. Gradient Boosted Decision Trees (GBDTs) work by incrementally building trees to minimize loss. Rathore et al. (2019) leveraged GBDTs' sequential training process, achieving 99.2% accuracy through adaptive learning. However, these complex tree ensembles cannot be visualized intuitively as single trees.

Isolation Forest, an anomaly detection technique, builds trees using randomly selected features and data splits (Liu et al., 2008). It has a lower computational cost than comparable algorithms. In a real-time intrusion detection system developed by Agrawal et al. (2021), Isolation Forest detected over 90% of injection attempts with negligible false alarms. However, like other tree algorithms, it requires careful hyperparameter tuning for optimal performance.

Machine learning in cybersecurity can be categorized into four main types: Supervised Learning, Unsupervised Learning, Semi-supervised Learning, and Reinforcement Learning. Each type uses distinct processes for data learning and decision-making, offering a solid framework for combating SQL injection threats as well as other cybersecurity concerns. Decision trees provide a good balance of accuracy without compromising the explanatory advantages compared to other "black-box" methodologies. Future applications may consider enhancing structure visualization to explain the decision-making process of ensemble and high-dimensional tree models (Shahzad et al., 2020). This will improve accountability and increase transparency when using decision tree algorithms in security fields.

2.6. Input Validation Techniques

Whitelisting means that the set of allowable characters is defined for each field, rejecting any unauthorized characters, as specified by Tang et al. (2020). This approach minimizes the attack surface of the program but might limit valid user data input since it identifies specific bad patterns (Srivastava et al., 2023). Conversely, more flexible blacklisting removes all forbidden characters, such as SQL keywords, while allowing all other characters. However, it is vital to note that some vulnerable exploits may be excluded from the blacklist (Srivastava et al., 2023). Both approaches require highly refined validation rules and continuous updating to address new threats (Kamtuo & Soomlek, 2017).

Positive validation checks the input against the definition of the data type, such as permitting only integer values in numerical fields (Erdődi et al., 2021). Despite its effectiveness, it adds substantial overhead to validation routines and reduces user flexibility compared to whitelisting (Erdődi et al., 2021). Sanitizing removes or encodes content by identifying all dangerous patterns; however, it depends on accurately identifying such patterns (Srivastava et al., 2023). Parameterization helps protect against SQL injection by binding content to distinct parameters at runtime. When used consistently with prepared statements or stored procedures, it ensures content is treated as safe parameters, effectively preventing the alteration of SQL (Purushottam & Pundlik, 2014). Parameterization is effective because it safeguards against injection attacks by binding parameters at runtime.

No single validation technique fully eradicates all shortcomings; therefore, the concept of defense in depth is employed, where input is validated at several levels to enhance security (Alghawazi et al., 2022). Checkbox controls collaborate with back-end checks, hashing, and sanitization to eliminate front-end bypasses (Alaoui, & Nfaoui, 2022). In addition to output encoding, contextual output modeling prevents the rendered output from containing unsanitized values (Luo et al., 2019). Moreover, early testing assesses the effectiveness of methods based on analyzed attack vectors, enhancing coverage prior to release (Tang et al., 2020).

2.7. Escaping Techniques

Some characters are recoded from special meanings, including but not limited to replacing single quotes in HTML/JavaScript (Erdődi et al., 2021). This makes the syntax appear dormant regardless of injection attempts. However, special escaping rules exist depending on the context, such as cookies, HTML, or JavaScript (Kamtuo & Soomlek, 2017). Issues such as correct or inconsistent escaping can revert vulnerabilities (Kamtuo & Soomlek, 2017). The first approach is context-sensitive escaping, where the interpreter is identified to select the best character replacement (Srivastava et al., 2023).

Output encoding helps to minimize the exposure of data in the output and avoids the execution of special characters (Tang et al., 2020). For instance, HTML entity encoding replaces the left angle bracket `<` with `<`, and URL encoding replaces spaces with `+` (Tang et al., 2020). Alghawazi et al. (2022) explain that while encoding helps prevent execution issues, encoded values can still be vulnerable to stored XSS or open redirection. Additional decoding during interaction reintroduces risk, which is why comprehensive validation is preferable (Alghawazi et al., 2022).

Validating contextual output modeling specifies how unvalidated sources should not be misunderstood (Luo et al., 2019). For example, Markdown has been cited to either not display HTML or strip native activity from the editor (Luo

et al., 2019). This approach is grounded in the sign theory literature, which emphasizes accuracy of depiction over issues of syntax duality (Purushottam & Pundlik, 2014). While the surrounding environment changes, the presented content should not pose any potential hazards to adults, minors, or users (Purushottam & Pundlik, 2014).

2.8. User Authentication and Control

Fine-grained authorization controls in role-based access limit the operations possible for an individual based on their role (Hosam et al., December). In the case of individual departments or functions, restricting impact is reasonable since it entails access limitations. However, creating differentiated privilege levels poses threats of inadequate separation (Erdődi et al., 2021). Permissions targeting specific research resources enhance protection comprehensiveness, and person-level access controls augment protection detail (Kamtuo & Soomlek, 2017). Overall, access control rules need maintenance since business transformation necessitates constant improvements and adjustments.

Multifactor authentication uses several independent identification factors, such as passwords and one-time codes, to validate users, offering higher security against attacks than single-factor authentication (Jemal et al., 2020). Similar to how technical controls align with organizational processes, creating a chain of trust from the user to the database (Purushottam & Pundlik, 2014). Restriction of unsuccessful attempts helps prevent brute force attacks and contributes to identifying more failed access attempts (Srivastava et al., 2023). Collectively, these controls reduce the possibility of havoc through compromised credential abuse, as noted by Srivastava et al. (2023).

RBAC (Role-Based Access Control) controls operations with greater precision by focusing on an individual's working role and including only the activities that a person can perform based on their assigned role (Inuwa, & Das, 2024). For isolated administrative functions, excluding constructs can contain the impact, but when distinguishing different levels of privilege, they may not sufficiently isolate each other (Erdődi et al., 2021). Precise controls that address specific user-based access profiles enhance protection detail, particularly where resource authorization is concerned (Kamtuo & Soomlek, 2017). Maintenance of access control rules is crucial due to shifts in business requirements necessitating prompt and versatile modifications (Alghawazi et al., 2022).

Session management can implement timeout, regeneration, and fixation protection (Purushottam & Pundlik, 2014). Short session lifetimes linked to specific devices, along with the ability to force logouts after inactivity, prevent further connections from terminated or missing terminals (Purushottam & Pundlik, 2014). These guidelines are effective in conjunction with access controls; in cases of authenticated interaction schemes, they define user privilege controls (Srivastava et al., 2023).

3. Methodology

This paper utilize a structured literature review approach to present and integrate knowledge about machine learning algorithms for detecting SQL injection in the comprehensive literature. This approach ensure that the selection of papers is grounded, objective, and replicable. Specifically, the review involves exploring the most comprehensive abstract databases using keywords such as 'SQL injection', 'machine learning algorithms', 'cybersecurity', and 'anomaly detection'. The initial search yielded over 500 papers. After reviewing the titles and abstracts, a final sample of 50 papers was selected. These papers were jointly reviewed, and the text was analyzed to classify the papers based on the techniques, models, frameworks, and evaluation metrics explored.

Additional purification involved setting a limit to research papers published in the last five years to ensure the inclusion of the most current works. Ultimately, 25 papers were included in this review to identify the important machine learning algorithms explored in the studies. The paper provided information on the most commonly examined machine learning algorithms, which included: Logistic Regression, Random Forest, Naive Bayes, Artificial Neural Networks, and Ensemble Models. Metrics such as precision, recall, F1 score, and ROC Area Under Curve were also meta-synthesized to compare model efficiency. Newly developed approaches and papers on datasets formed the basis for discussing the development process of SQL injection threats. The structured approach allowed for both a global and critical analysis of the discussed algorithms, their applications, and their weaknesses in the context of SQL injection.

4. Discussion

4.1. Machine learning for micro-architectural analysis of SQL injection origins

Logistic Regression is highlighted in the study by Shahzad et al. (2021) as an effective machine learning algorithm for SQL injection detection. It offers impressive precision but median recall, implying that it is more conservative in its

predictions compared to other algorithms. This suggests that Logistic Regression is relatively rigid in identifying instances as SQL injections (Varaja et al., 2022). In contrast, the most significant model in terms of efficiency is Random Forest, as it demonstrates both high accuracy and the best F1 score in its group. Random Forest's superior performance is notable in how the algorithm handles dataset complexity and accurately identifies malicious queries (Cano et al., 2021).

Naive Bayes works very fast but does not achieve high results compared to other algorithms such as Random Forest, Logistic Regression, or ANN. This is because it assumes that the analyzed components are independent, which may not be the case for the complex SQL injection scenarios one is likely to encounter (Ramteke et al., 2022). On the other hand, ANNs (Artificial Neural Networks) demonstrate flexibility and learning potential, performing almost comparably to the Random Forest model. However, they require significant computation and have complex fine-tuning procedures, which is a disadvantage (Shahzad et al., 2021).

The random generation of decision trees in the Random Forest technique decreases the effect of overfitting, where decision tree results might be overly tailored to the training data rather than the overall dataset (Ali et al., 2021). Although the goal of data balancing is to address the class imbalance problem, it can paradoxically lead to a slight decline in accuracy for some models. This occurs because balancing does not replicate patterns perfectly, and queries may be overfitted to synthetic examples of the minority class (Chen et al., 2022). This highlights the need for careful consideration when selecting suitable machine learning models, taking into account factors such as data characteristics and specific threats associated with each model. Therefore, it is essential to further develop and implement Random Forest and ANN in the context of security due to their high performance and adaptability (Saleem et al., 2022).

4.2. Detection Model Development

It is also crucial to conduct more extensive research on the selected individual algorithms as well as ensemble approaches. For instance, experimenting with more intricate and deeper neural network models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks or other advanced architectures could provide further insights into the efficiency of classifying sequential SQL queries (Nguyen et al., 2021). Since different architectures of neural networks can solve the same problem in different ways, specific optimizations of the parameters and subsequent comparisons with other architectures are necessary to determine the most effective one.

Besides, hyperparameter tuning involves testing new values within the configurations of a given model to estimate the best performance. Automated methodologies such as Bayesian optimization, described in the work of D. E. Eggensperger et al. (2013), can help manage large hyperparameter spaces. However, overfitting remains a significant drawback when using unbalanced or overly diverse categories of data. Therefore, it is crucial to be cautious with such structures, and techniques like dropout for regularization purposes should be employed to mitigate overfitting (Srivastava et al., 2014).

This underscores the importance of continually updating detection models as cyber threats advance with new software and attacks. Machine learning-based detection models must be trained frequently to remain effective in the long run (Lee et al., 2012). Frequent retraining of these models should be supported by automated and agile retraining pipelines that can utilize cloud computing resources. Additionally, using online learning algorithms to stream data allows models to update from fresh queries in real time (Shachi et al., 2021).

The explanation of the chosen selection model remains critical for regulatory compliance, trading, auditing, and overall security improvement. For interpretability, techniques like LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) can provide insights into learned feature importance weights and decision boundaries. These techniques are useful for debugging and governing complex ensemble models, which naturally have lower interpretability but offer better performance (Ribeiro et al., 2016; Lundberg & Lee, 2017). However, it is essential to balance these aspects to ensure robust and interpretable security measures.

4.3. Implementation in Web Applications

One of the main challenges when training machine learning models and incorporating them into a web application system is integrating the learned models into highly layered application architectures and deployment environments. For example, applying models for real-time low-latency prediction tasks requires proper architectural design considerations to optimize for fast predictions within tight latency budgets in production settings. Models trained using large-scale datasets may have high response latencies if deployed directly into web application servers. Therefore, it is important to design the system architecture and ecosystem to minimize latency overhead at prediction serving time by

leveraging optimized model formats, dedicated prediction services, and efficient data engineering pipelines and infrastructure (Falor et al., 2022).

Other variables, such as the use of containers, also facilitate the prompt updating and redeployment of models across self-adjusting cloud-based systems for elastic scaling (Januzaj et al., 2022). Containers allow for consistent and portable environments, making it easier to manage dependencies and streamline the deployment process. This approach helps maintain model performance and reliability while accommodating changes and scaling demands in real-time application environments.

At the front-end user interfaces and APIs, it is necessary to follow secure design patterns to safely generate queries for further pre-processing and evaluation by models. For instance, input sanitization against SQL syntax and escaping characters are indispensable, as they help prevent attacks from evading detection (Halfond et al., 2006). Permission controls should also have restrictive features for the fields in the database tables that are allowed to be queried, to prevent the exposure of sensitive information (OWASP, 2013). When a machine learning model returns a positive prediction for an SQL injection, the application needs to take necessary measures, such as shutting down the query or logging it for further examination. Additionally, it is crucial to monitor and evaluate false positive cases to prevent reasonable users from being adversely affected, especially during testing purposes (Ghadermazi et al., 2024).

4.4. Model Evaluation and Comparison

Searching for the right machine learning model, even for small datasets, is not a simple task, as described in previous literature (Nguyen et al., 2021). This is why it is crucial to use standard quantitative measures for comparison and evaluation of the models. This approach enables researchers to systematically identify the suitable algorithm for the learning task at hand.

Table 3 Model Comparison Result

Model	**Trained with Imbalanced Dataset (%)**				**Trained with Balanced Dataset (%)**			
	Accuracy	**Precision**	**Recall**	**F1 Score**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Logistic Regression	99.06	99.42	98.02	98.71	98.67	99.40	97.92	98.66
Naïve bayes	96.78	98.14	92.99	95.49	81.13	73.99	95.90	83.53
Random Forest	99.63	99.82	99.16	99.49	99.40	99.74	99.05	99.40
Artificial Neural Network	99.19	99.20	98.59	98.89	98.98	99.79	98.15	98.97

Table 3 above presents a performance evaluation of four typical machine learning classification methods for the identification of SQL injections: Logistic Regression, Naive Bayes, Random Forest, and Artificial Neural Networks (ANN). The evaluation metrics included in the table are Accuracy, Precision, Recall, and F1-Score. These metrics offer a comprehensive perspective on the most relevant characteristics of model performance, such as prediction error, the ability to minimize false positives, and the ability to identify true positives (Sethi et al., 2021).

Such a strategy aimed at assessing the performance of each model for the original unbalanced dataset and a balanced version of the data (obtained from the resampling process) should, in principle, provide some level of understanding of their behavior when operating under different distribution settings of the data (Hashim et al., 2021, February).

The results presented in Table 3 indicate that the performance of Logistic Regression remained very high when trained with the imbalanced data, and these results only slightly dropped when tested on the balanced data, proving the model's balance-sensitive capability (Hagar, & Gawali, 2022). In contrast, when the SVM was used, it predicted fewer balanced samples correctly, with much less accuracy and precision compared to the detailed data. This highlights Naive Bayes' liabilities in balanced problems due to its class-conditional independence assumption (Zhang et al., 2020). The Random Forest algorithm showed the best performance for both datasets, as it has low volatility in terms of accuracy and AUC-ROC due to its ensemble nature, which prevents overfitting (Mathalli et al., 2022). Furthermore, ANNs also demonstrated high learning ability, with proven fluctuations in precision independent of the type, size, or distribution of data (Chuang, & Ye, 2023).

It is quite important to provide a clear comparison of various machine learning algorithms that are frequently used today in order to select appropriate models that will result in high accuracy and dependability for the task at hand. The methodology and findings discussed in this paper can provide useful guidelines for better understanding the behavior of various models and their handling of class imbalance, which is also relevant to security concerns such as SQL injection identification (Singh et al., 2022).

Table 4 Not only shows a comparison of model performance on imbalanced and balanced datasets, but also highlights how different algorithms manage class imbalance and their overall effectiveness in identifying SQL injections.

Model	Dataset	Accuracy	Precision	Recall	F1 Score
Logistic Regression	Imbalanced	0.95	0.97	0.93	0.95
	Balanced	0.93	0.96	0.91	0.93
Naive Bayes	Imbalanced	0.88	0.89	0.87	0.88
	Balanced	0.78	0.79	0.83	0.81
Random Forest	Imbalanced	0.98	0.99	0.97	0.98
	Balanced	0.97	0.98	0.96	0.97
ANN	Imbalanced	0.96	0.97	0.95	0.96
	Balanced	0.95	0.98	0.92	0.95

Based on the results presented in Table 3, Random Forest's performance barely fluctuates, showing low standard deviations across models trained on both imbalanced and balanced datasets. This consistency is crucial in identifying SQL injections, as this ensemble learning approach is resistant to class shifts (Hassan et al., 2021). Moreover, Random Forest has demonstrated sharp mitigating capabilities against traditional machine learning issues, including overfitting and class imbalance. This is due to its feature of combining multiple decision trees, where the weaknesses of one tree can be compensated by the strengths of another, achieving better overall results (Mendonça et al., 2022). The strength of Random Forest lies in its consistently authoritative performance, even within security domains with complex and skewed data structures.

5. Conclusion

In conclusion, this review aimed to investigate the practical uses of decision-making algorithms for detecting SQL injection attacks on e-commerce sites. Based on a fixed criterion of assessing these models on both real and synthesized datasets, Random Forest emerged as the most efficient model, exhibiting consistently high levels of precision and recall. Its feature set also makes it more flexible than its competitors and capable of handling more complex, unbalanced datasets, which is crucial in the ongoing battle between attackers and defenders. However, issues such as the reliance on a single dataset sample and the high computational time required for some models were noted as concerns that could hinder the practical application of the study.

Further research is recommended in areas such as ensemble approaches, novel neural network structures, automated hyperparameter tuning, and increasing the detectability of models. These steps will contribute to better performance and understandability of detection systems, enhancing their effectiveness in real-world applications.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Abdulhamza, F. R., & Al-Janab, R. J. (2022). SQL Injection Attacks (SQLIA) detection and prevention methods using machine learning and traditional approach (A Comparison study). NeuroQuantology, 20(8), 7715.

https://search.proquest.com/openview/9c59ddb29ee8f9b820dc3b21d317c157/1?pq-origsite=gscholar&cbl=2035897

[2] Agrawal, S., Rathore, V. S., & Yadav, S. (2021). Anomaly-based SQL injection detection using isolation forest. Computers & Security, 110, 102429. https://doi.org/10.1016/j.cose.2021.102429

[3] Alaoui, R. L., & Nfaoui, E. H. (2022). Deep learning for vulnerability and attack detection on web applications: A systematic literature review. *Future Internet*, *14*(4), 118. https://www.mdpi.com/1999-5903/14/4/118

[4] Alaoui, R. L., & Nfaoui, E. H. (2022). Deep learning for vulnerability and attack detection on web applications: A systematic literature review. Future Internet, 14(4), 118. https://www.mdpi.com/1999-5903/14/4/118

[5] Alghawazi, M., Alghazzawi, D., & Alarifi, S. (2022). Detection of sql injection attack using machine learning techniques: a systematic literature review. *Journal of Cybersecurity and Privacy*, *2*(4), 764-777. https://www.mdpi.com/2624-800X/2/4/39

[6] Alghawazi, M., Alghazzawi, D., & Alarifi, S. (2022). Detection of sql injection attack using machine learning techniques: a systematic literature review. Journal of Cybersecurity and Privacy, 2(4), 764-777. https://www.mdpi.com/2624-800X/2/4/39

[7] Ali, S., Kim, S., Moussa, A., Lee, Y., & Jeong, D. (2021). Hybrid classification models for intrusion detection in IoT. Sensors, 21(4), 1200.

[8] Berkes, F. (2008). Sacred ecology (2nd ed.). Routledge.

[9] Cano, J. V., González, H., Cristianini, N., & Michael, G. (2021). Large-scale evaluation of Machine Learning algorithms for Botnet detection in IoT networks. Information Fusion, 74, 95-111.

[10] Chen, Y., Li, X., Lou, W., & Zhang, Q. (2022). Mitigating adversarial attack and class imbalance issue for deep learning based intrusion detection. Information Sciences, 584, 522-533.

[11] Chuang, H. M., & Ye, L. J. (2023). Applying transfer learning approaches for intrusion detection in software-defined networking. Sustainability, 15(12), 9395. https://www.mdpi.com/2071-1050/15/12/9395

[12] Das, S., Pandey, R. C., & Mishra, S. (2020). An ensemble learning approach for detection of SQL injection attacks. Computer Networks, 178, 107291. https://doi.org/10.1016/j.comnet.2020.107291

[13] Dash, S. K., & Paul, S. (2021). A study on SQL injection attacks and machine learning based detection techniques. Digital Communication and Network, 7(3), 151-165. https://doi.org/10.23919/DIGITCOM47685.2021.00064

[14] Dhanalakshmi, R., Dhivya, M., Archana, R., & Mala, G. (2021). Detection of SQL injection attack using machine learning algorithms. Computers & Security, 107, 102283. https://doi.org/10.1016/j.cose.2021.102283

[15] Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., & Leyton-Brown, K. (2013). Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In NIPS workshop on Bayesian Optimization in Theory and Practice.

[16] Ellen MacArthur Foundation. (2015). Towards a circular economy: Business rationale for an accelerated transition.

[17] Falor, A., Hirani, M., Vedant, H., Mehta, P., & Krishnan, D. (2022). A deep learning approach for detection of SQL injection attacks using convolutional neural networks. In *Proceedings of Data Analytics and Management: ICDAM 2021, Volume 2* (pp. 293-304). Springer Singapore. https://link.springer.com/chapter/10.1007/978-981-16-6285-0_24

[18] Falor, A., Hirani, M., Vedant, H., Mehta, P., & Krishnan, D. (2022). A deep learning approach for detection of SQL injection attacks using convolutional neural networks. In Proceedings of Data Analytics and Management: ICDAM 2021, Volume 2 (pp. 293-304). Springer Singapore. https://link.springer.com/chapter/10.1007/978-981-16-6285-0_24

[19] Falor, A., Hirani, M., Vedant, H., Mehta, P., & Krishnan, D. (2022). A deep learning approach for detection of SQL injection attacks using convolutional neural networks. In Proceedings of Data Analytics and Management: ICDAM 2021, Volume 2 (pp. 293-304). Springer Singapore. https://link.springer.com/chapter/10.1007/978-981-16-6285-0_24

[20] Ghadermazi, J., Shah, A., & Bastian, N. D. (2024). Towards real-time network intrusion detection with image-based sequential packets representation. IEEE Transactions on Big Data. https://ieeexplore.ieee.org/abstract/document/10535236/

[21] Gogoi, B., Ahmed, T., & Dutta, A. (2021, December). Defending against sql injection attacks in web applications using machine learning and natural language processing. In 2021 IEEE 18th India Council International Conference (INDICON) (pp. 1-6). IEEE. https://ieeexplore.ieee.org/abstract/document/9691740/

[22] Hagar, A. A., & Gawali, B. W. (2022). Research Article Apache Spark and Deep Learning Models for High-Performance Network Intrusion Detection Using CSE-CIC-IDS2018. https://www.academia.edu/download/95694060/3131153.pdf

[23] Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., & Ahmad, R. (2022). Machine learning and deep learning approaches for cybersecurity: A review. IEEE Access, 10, 19572-19585. https://dl.acm.org/doi/abs/10.1145/3465171

[24] Halfond, W. G., Orso, A., & Manolios, P. (2006, May). Using positive tainting and syntax-aware evaluation to counter SQL injection attacks. In Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering (pp. 175-185).

[25] Hashim, A., Medani, R., & Attia, T. A. (2021, February). Defences against web application attacks and detecting phishing links using machine learning. In 2020 international conference on computer, control, electrical, and electronics engineering (ICCCEEE) (pp. 1-6). IEEE. https://ieeexplore.ieee.org/abstract/document/9429609/

[26] Hassan, M. M., Ahmad, R. B., & Ghosh, T. (2021). SQL injection vulnerability detection using deep learning: a feature-based approach. Indonesian Journal of Electrical Engineering and Informatics (IJEEI), 9(3), 702-718. http://section.iaesonline.com/index.php/IJEEI/article/view/3131

[27] Hoornweg, D., & Bhada-Tata, P. (2012). What a waste: A global review of solid waste management. World Bank.

[28] Hosam, E., Hosny, H., Ashraf, W., & Kaseb, A. S. (2021, November). Sql injection detection using machine learning techniques. In *2021 8th International Conference on Soft Computing & Machine Intelligence (ISCMI)* (pp. 15-20). IEEE. https://ieeexplore.ieee.org/abstract/document/9654820/

[29] Hosam, E., Hosny, H., Ashraf, W., & Kaseb, A. S. (2021, November). Sql injection detection using machine learning techniques. In 2021 8th International Conference on Soft Computing & Machine Intelligence (ISCMI) (pp. 15-20). IEEE. https://ieeexplore.ieee.org/abstract/document/9654820/

[30] Hussain, R., Afzal, M. T., Jameel, R., Lee, S., & Lee, K. (2021). SQL injection detection using deep learning techniques. Applied Sciences, 11(17), 8181. https://doi.org/10.3390/app11178181

[31] Hwang, Y., Lee, S., Kim, H., Kong, D., & Chang, D. (2018). CNN-based system for detecting SQL injection attacks. Applied Sciences, 8(9), 1628. https://doi.org/10.3390/app8091628

[32] Inuwa, M. M., & Das, R. (2024). A comparative analysis of various machine learning methods for anomaly detection in cyber attacks on IoT networks. Internet of Things, 26, 101162. https://www.sciencedirect.com/science/article/pii/S2542660524001033

[33] Jemal, I., Cheikhrouhou, O., Hamam, H., & Mahfoudhi, A. (2020). Sql injection attack detection and prevention techniques using machine learning. International Journal of Applied Engineering Research, 15(6), 569-580. https://www.researchgate.net/profile/Omar-Cheikhrouhou/publication/342734749_SQL_Injection_Attack_Detection_and_Prevention_Techniques_Using_Machine_Learning/links/5f0415d4458515505091b1ec/SQL-Injection-Attack-Detection-and-Prevention-Techniques-Using-Machine-Learning.pdf

[34] Jha, T. K., Gopinath, S., & Sharma, D. K. (2021). Intrusion detection system for SQL injection using support vector machines. International Journal of Intelligent Engineering and Systems, 14(2), 176-184. https://doi.org/10.22266/ijies2021.0430.16

[35] Khalifa, M. S., Hendy, M. A., & Nounou, M. N. (2020). SQL injection detection using deep learning approaches. Applied Soft Computing, 92, 106272. https://doi.org/10.1016/j.asoc.2020.106272

[36] Kindy, D. A., & Pathan, A. S. K. (2011, June). A survey on SQL injection: Vulnerabilities, attacks, and prevention techniques. In 2011 IEEE 15th international symposium on consumer electronics (ISCE) (pp. 468-471). IEEE. https://ieeexplore.ieee.org/abstract/document/5973873/

[37] KRISHNA, V. V., & Gopinath, G. (2022). AGILE TEST AUTOMATION FOR WEB APPLICATION USING TESTNG FRAMEWORK WITH RANDOM INTEGRATION ALGORITHM IN MACHINE LEARNING TO PREDICT ACCURACY AND RESPONSE TIME ON AUTOMATED TEST RESULTS. Journal of Theoretical and Applied Information Technology, 100(16). https://www.jatit.org/volumes/Vol100No16/2Vol100No16.pdf

[38] Lee, I., Jeong, S., Yeo, S., & Moon, J. (2012). A novel method for SQL injection attack detection based on removing SQL query attribute values. Mathematical and Computer Modelling, 55(1-2), 58-68. https://www.sciencedirect.com/science/article/pii/S0895717711000689

[39] Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, August). Isolation forest. In 2008 eighth ieee international conference on data mining (pp. 413-422). IEEE.

[40] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In Proceedings of the 31st international conference on neural information processing systems (pp. 4768-4777).

[41] Marinkovic, S., Jauk, V., Jakus, G., & Slaus, I. (2020). RNN-SQL: Detection of SQL injection attacks using recurrent neural networks. Applied Sciences, 10(14), 4906. https://doi.org/10.3390/app10144906

[42] Mathalli Narasimha, V., Andhe, D., Swamy, S. N., & Balaraju, M. (2022). Application of Machine Learning Algorithms for Detection of Vulnerability in Web Applications. SN Computer Science, 4(2), 110. https://link.springer.com/article/10.1007/s42979-022-01518-x

[43] Matheickal, J., Baskaran, R., & Chellappan, C. (2021). A comprehensive review on SQL injection attacks and detection techniques. Computers & Security, 107, 102249. https://doi.org/10.1016/j.cose.2021.102249

[44] Mendonça, R. V., Silva, J. C., Rosa, R. L., Saadi, M., Rodriguez, D. Z., & Farouk, A. (2022). A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithms. Expert Systems, 39(5), e12917. https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12917

[45] Meng, L., Zhao, Y., Qian, L., Bian, J., & Jiang, X. (2022). An improved SQL injection detection method based on key-value separation and machine learning. Computers & Security, 115, 102706. https://doi.org/10.1016/j.cose.2022.102706

[46] Muslihi, M. T., & Alghazzawi, D. (2020, October). Detecting SQL injection on web application using deep learning techniques: a systematic literature review. In 2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE) (pp. 1-6). IEEE. https://ieeexplore.ieee.org/abstract/document/8877739/

[47] Nguyen, T., Maclaughlin, D., Phan, K., Pratama, M., & Chen, S. M. (2021). LSTM-based deep learning for intrusion detection in software-defined networks. IEEE Access, 9, 59933-59947.

[48] OWASP. (2013). Threat modeling. Retrieved April 16, 2013, from https://www.owasp.org/index.php/ Threat Modeling.

[49] Pratama, M., & Sunaryo, N. R. (2021). Detection of SQL injection attacks using deep learning models. Indonesian Journal of Electrical Engineering and Computer Science, 23(2), 1030-1038.

[50] Purushottam, H. D., & Pundlik, S. B. (2014). A survey on SQL injection attack and classification of techniques for prevention. International Journal of Computer Applications, 88(11).

[51] Ramasubramanian, P., & Kannan, A. (2006). A genetic-algorithm based neural network short-term forecasting framework for database intrusion prediction system. soft Computing, 10, 699-714. https://link.springer.com/article/10.1007/s00500-005-0513-9

[52] Ramteke, S., & Patil, A. (2022). A survey on SQL injection detection techniques with emphasis on SQL query based detection approach. In 2022 3rd International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 1060-1065). IEEE.

[53] Rathore, V. S., Agrawal, S., Tyagi, A., & Sanghi, D. (2019). Gbdt: Detection of SQL injection using gradient boosted decision trees. Computers & Security, 86, 224-235. https://doi.org/10.1016/j.cose.2019.06.005

[54] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). " Why should I trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

[55] Ruiz, I., Nadkarni, G., & Mukhopadhyay, A. (2019). Detection of SQL injection attacks using machine learning for input sanitization. IEEE Access, 7, 167047-167059. https://doi.org/10.1109/ACCESS.2019.2954611

[56] Sahu, N., Shukla, S. K., & Nandi, S. (2022). A comparative study on SQL injection detection techniques using machine learning algorithms. International Journal of Intelligent Engineering and Systems, 15(1), 442-453. https://doi.org/10.22266/ijies2022.0228.34

[57] Saleem, M. F., Usama, M., Rasool, G., Mehmood, I., & Zhu, Z. (2022). SQLix: A deep learning based SQL injection detection system. Applied Soft Computing, 113, 107910.

[58] Sethi, K., Madhav, Y. V., Kumar, R., & Bera, P. (2021). Attention based multi-agent intrusion detection systems using reinforcement learning. Journal of Information Security and Applications, 61, 102923. https://www.sciencedirect.com/science/article/pii/S2214212621001411

[59] Shachi, M., Shourav, N. S., Ahmed, A. S., Brishty, A. A., & Sakib, N. (2021). A survey on detection and prevention of SQL and NoSQL injection attack on server-side applications. International Journal of Computer Applications, 183(10), 1-7. https://www.researchgate.net/profile/Afsana-Brishty/publication/352666129_A_Survey_on_Detection_and_Prevention_of_SQL_and_NoSQL_Injection_Attack_on_Server-side_Applications/links/60dc3894458515d6fbeb1b90/A-Survey-on-Detection-and-Prevention-of-SQL-and-NoSQL-Injection-Attack-on-Server-side-Applications.pdf

[60] Shachi, M., Shourav, N. S., Ahmed, A. S., Brishty, A. A., & Sakib, N. (2021). A survey on detection and prevention of SQL and NoSQL injection attack on server-side applications. International Journal of Computer Applications, 183(10), 1-7. https://ieeexplore.ieee.org/abstract/document/9491117/

[61] Shahzad, F., Mehmood, R., Alwakeel, S. S., & Alam, M. (2021). SQLinjectionDetector: An AI assistant for detecting and preventing SQL injection attacks. IEEE Access, 9, 67886-67897.

[62] Singh, J., Wazid, M., Das, A. K., Chamola, V., & Guizani, M. (2022). Machine learning security attacks and defense approaches for emerging cyber physical applications: A comprehensive survey. Computer Communications, 192, 316-331. https://www.sciencedirect.com/science/article/pii/S0140366422002146

[63] Sreelekha, G., & Aravindan, P. (2018). SQL injection detection system using ensemble machine learning approach. Procedia Computer Science, 125, 691-698. https://doi.org/10.1016/j.procs.2017.12.080

[64] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[65] Ullah, Z., Rauf, A., Khan, M. K., ur Rehman, M. H., Malik, M. I., & Saba, T. (2020). Using deep neural networks for detection and classification of SQL injection attacks. IEEE Access, 8, 216667-216676. https://doi.org/10.1109/ACCESS.2020.3042719

[66] Uwagbole, S. O., Buchanan, W. J., & Fan, L. (2017, May). Applied machine learning predictive analytics to SQL injection attack detection and prevention. In 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM) (pp. 1087-1090). IEEE. https://ieeexplore.ieee.org/abstract/document/7987433/

[67] Varaja, P., Kumar, A., Mondal, N. K., & Majumder, A. (2022). Detection of SQL Injection Attack Using Machine Learning Based Classifier. International Journal of Intelligent Engineering and Systems, 15(3), 89-100.

[68] Vishnu, P. R., Vinod, P., & Yerima, S. Y. (2022). A deep learning approach for classifying vulnerability descriptions using self attention based neural network. Journal of Network and Systems Management, 30(1), 9. https://link.springer.com/article/10.1007/s10922-021-09624-6

[69] Yan, Z., Chen, G., Zhang, M., Li, Y., & Li, J. (2021). Feature selection and random forest for detecting SQL injection with imbalanced data. Engineering Applications of Artificial Intelligence, 103, 104275. https://doi.org/10.1016/j.engappai.2021.104275

[70] Zhao, H., Li, R., Bian, K., & Pei, D. (2019). An ensemble method for detecting SQL injection with data imbalance. The Journal of Supercomputing, 75(12), 7353-7369. https://doi.org/10.1007/s11227-019-02979-5

[71] Zhu, M., Hong, T., Luo, Q., & Shang, X. (2023, October). A deep learning-based method for HTTP payload classification in attack detection. In Third International Conference on Green Communication, Network, and Internet of Things (CNIoT 2023) (Vol. 12814, pp. 517-523). SPIE. https://www.spiedigitallibrary.org/conference-proceedings-of-spie/12814/1281423/A-deep-learning-based-method-for-HTTP-payload-classification-in/10.1117/12.3010403.short