Injectiq: ML-Powered SQL Injection Detection For Saas Applications

Rachana N, Kshama N, Sukhi S Venki, Prajwal R ,Thanuja N

Department of Computer Science
Bangalore Institute of Technology
Bangalore, India

***Abstract*—** Web applications today are increasingly targeted by SQL Injection (SQLi) attacks, one of the most prevalent and damaging vulnerabilities affecting online systems. Traditional security mechanisms such as signature-based filters, firewalls, and manual validation provide incomplete protection against modern SQLi payloads that use obfuscation, encoding, and adversarial variations to bypass static rules. This work proposes a hybrid SQL Injection Detection System that integrates regular-expression–based preprocessing, deep-learning–driven feature extraction using a 1D Convolutional Neural Network (CNN), and ensemble classification using Random Forest. Multiple public datasets—including SQLiV3, CSIC HTTP dataset, and community payload repositories—are combined to construct a diverse and realistic dataset. Queries are tokenized, padded, and transformed into feature vectors, while handcrafted numerical features capture structural attributes of SQL inputs. The hybrid model performs real-time classification and distinguishes malicious queries from benign user inputs with high accuracy. Experimental results demonstrate improved detection performance compared to standalone ML or DL models. The system highlights the potential of lightweight hybrid architectures in strengthening application-layer security and mitigating SQLi threats in practical web environments.

Index Terms— SQL Injection Detection, Web Security, CNN Feature Extraction, Random Forest Classification, Hybrid Model, Input Validation, Machine Learning.

I. INTRODUCTION

SQL Injection remains one of the most critical security risks in modern web applications. It enables attackers to manipulate backend database queries through malicious user inputs, leading to unauthorized data access, account compromise, privilege escalation, and in severe cases, full database disclosure. As web platforms expand to support complex SaaS architectures and user-driven functionalities, the attack surface for SQLi continues to grow. The ease of exploitation combined with its high impact has made SQLi a persistent challenge in the OWASP Top 10 for over a decade.

Traditional defenses—such as regular expressions, Web Application Firewalls (WAFs), and manual sanitization—offer partial protection but suffer from several limitations. Static patterns fail against adversarial payloads that use encoding, comment injection, case manipulation, or logical restructuring. WAFs also depend heavily on signature updates and often generate false positives in dynamic, user-driven web environments. As attackers develop increasingly obfuscated SQLi vectors, purely rule-based approaches become insufficient.

Machine learning and deep learning have emerged as promising alternatives that analyze input patterns beyond simple keyword matching. CNNs capture local token dependencies within queries, while statistical models learn structural patterns. However, deep models alone may misclassify noisy or uncommon queries, and ML models require strong feature engineering. To address these issues, we present a hybrid SQL Injection Detection System that combines the strengths of both approaches.

The proposed system employs a two-stage detection process. A lightweight regex filter performs early elimination of clearly malicious queries. More complex inputs are processed using a CNN-based feature extractor alongside handcrafted numeric features that capture query length, digit count, special character density, and SQL keyword presence. These features are fed into a Random Forest classifier that provides robust decision boundaries and improves generalization across diverse SQLi patterns. This hybrid architecture aims to deliver high accuracy while remaining computationally efficient for real-time web deployment.

II. LITERATURE REVIEW

SQL Injection detection systems have evolved significantly over the years, beginning with rule-based filters and gradually incorporating machine learning and deep learning techniques. Traditional approaches use keyword matching, blacklists, and syntactic patterns to identify suspicious queries. While these systems are simple and fast, they struggle against obfuscated payloads and require frequent rule updates.

Machine learning–based SQLi detection has gained popularity due to its ability to generalize beyond predefined patterns. Researchers have explored algorithms such as Support Vector Machines (SVM), Naïve Bayes, Logistic Regression, and Random Forests using manually engineered features including token frequency and query structure. These models improve detection accuracy but depend heavily on feature selection and may fail to capture semantic relationships within inputs.

Deep learning approaches—including CNNs, LSTMs, and transformer-based models—automatically learn hierarchical representations of SQL queries. CNNs have been particularly effective in capturing local token patterns and detecting subtle variations in malicious payloads. However, deep networks require large, well-labeled datasets and may underperform when exposed to noisy or adversarial inputs.

Hybrid architectures combining rule-based filters with ML or DL models have shown strong performance in recent studies. These systems benefit from the efficiency of regex-based filtering while leveraging the predictive strength of ML/DL classifiers. Some works integrate CNN embeddings with traditional classifiers like SVM or Random Forest to enhance robustness. However, many existing models rely on synthetic datasets or evaluate only a narrow set of SQLi payloads, limiting their applicability in real-world environments.

A gap remains in the availability of multi-source datasets, methods that blend handcrafted numeric features with deep text embeddings, and systems optimized for production-level performance. The proposed hybrid model addresses these gaps by incorporating regex filtering, CNN-based feature extraction, and Random Forest classification within a unified detection pipeline

III. PROPOSED SYSTEM WITH ARCHITECTURE

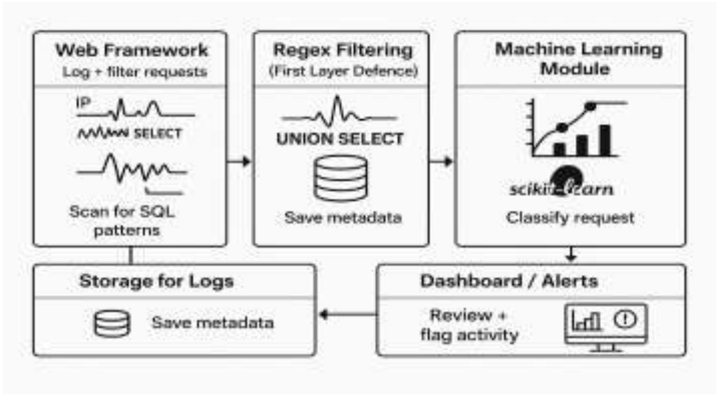

Fig 1: System Architecture

The proposed solution consists of four primary modules:

- Web Framework
- Regex Filtering
- Machine Learning Module
- Storage for Logs
- Dashboard/ Alerts

1. API Gateway and Input Handling Layer

This module serves as the system's primary entry point using FastAPI to receive all HTTP requests. It extracts payloads (query parameters, JSON, form data) along with metadata such as timestamps, IP addresses, and user-agent details. Minimal preprocessing ensures low latency. Before forwarding requests, it invokes the Regex Filtering layer for initial threat screening. All incoming queries and metadata are logged for auditing, monitoring, and retraining. Key endpoints include /detect for SQLi detection and /metrics for system health.

2. Regex-Based Preliminary Filtering Module

This deterministic layer performs the first stage of security analysis using curated regular expression signatures for common SQLi patterns (UNION-based, tautologies, comment injections, stacked queries, destructive commands). It generates a binary regex_match and a continuous regex_score based on detected severity. Payloads above a critical threshold

(≥0.9) are immediately blocked, while others proceed to machine-learning components with the score included as a feature. This reduces computational load and improves early-stage detection accuracy.

3. Preprocessing, Normalization, and Tokenization Pipeline

All incoming text is normalized by lowercasing, collapsing whitespace, removing noise, and abstracting literals. URL-encoded and HTML-escaped characters are converted into canonical representation. The tokenization stage converts the cleaned query into SQL-aware tokens mapped to a fixed vocabulary, including <UNK> handling for unseen tokens. After padding or truncation, these structured sequences serve as input to the CNN-based feature extractor.

4. CNN + Engineered Feature Extraction Module

This module combines deep semantic feature extraction with interpretable numeric characteristics. CNN Component: A 1-D convolutional network processes token embeddings using multiple kernel sizes to capture n-gram patterns such as UNION SELECT or OR 1=1. Global max-pooling creates fixed-size semantic embeddings summarizing SQL structure and contextual behavior.
Engineered Features: In parallel, handcrafted metrics such as query length, special character density, SQL keyword frequency, entropy, token count, nested query depth, comment density, and the regex_score are computed. All numeric features are standardized to create a unified representation.

5. Random Forest Classification & Decision Engine Module

The Random Forest classifier receives the concatenated vector of CNN embeddings and engineered numeric features. It outputs both a predicted label (benign/malicious) and a probability score of SQLi. RF is selected for its robustness to noisy, mixed-type data and strong generalization ability. Decisions are made by applying threshold-based policies: high-probability malicious queries are blocked and alerted, medium-risk queries may undergo additional verification, and low-risk traffic is allowed. Hyperparameters such as estimator count, tree depth, and class weighting are optimized to minimize false positives.

6. Deployment, Logging & Monitoring

The full hybrid pipeline is deployed as a scalable microservice optimized for low-latency inference. CNN components may be exported to ONNX, and RF models remain memory-resident for fast execution. Comprehensive logs record payloads, regex scores, model outputs, and decisions. Monitoring dashboards visualize attack trends, hotspot IPs, regex match frequencies, and model performance metrics. Misclassifications (false positives/negatives) feed into a retraining buffer, enabling continuous adaptation to evolving SQL injection techniques.

IV. METHODOLOGY

Tools and Technologies used:

Programming Languages: Python
Web Framework & API Tools: React (JSX), Postman API
Machine Learning Libraries: Scikit-learn, TensorFlow, Pandas & NumPy
Database: PostgreSQL
Development & Testing Tools: Jupyter Notebook , VS Code

This section provides a brief overview of the core components that form the hybrid SQL injection detection system. The Regex module performs fast, rule-based screening using curated SQLi patterns and assigns a severity score for early threat elimination. The CNN extractor learns semantic and structural patterns from tokenized queries through 1-D convolutions, while numeric and engineered features capture statistical cues such as length, entropy, keyword counts, and nesting depth. These features and CNN embeddings are fused in a Random Forest classifier, forming the hybrid RF+CNN pipeline that delivers robust detection even against obfuscated attacks. The system also incorporates a request logging mechanism that preserves input traces for auditing and model retraining purposes. Additionally, an alerting framework notifies administrators in real time about suspicious activity, enabling prompt mitigation. Evaluation is conducted using accuracy, precision, recall, F1-score, ROC-AUC, latency, and throughput to ensure reliable performance in real-time SaaS environments. Extensive testing with diverse attack vectors, including encoded and multi-layered payloads, demonstrates the system's adaptability and resilience against emerging SQL injection techniques.

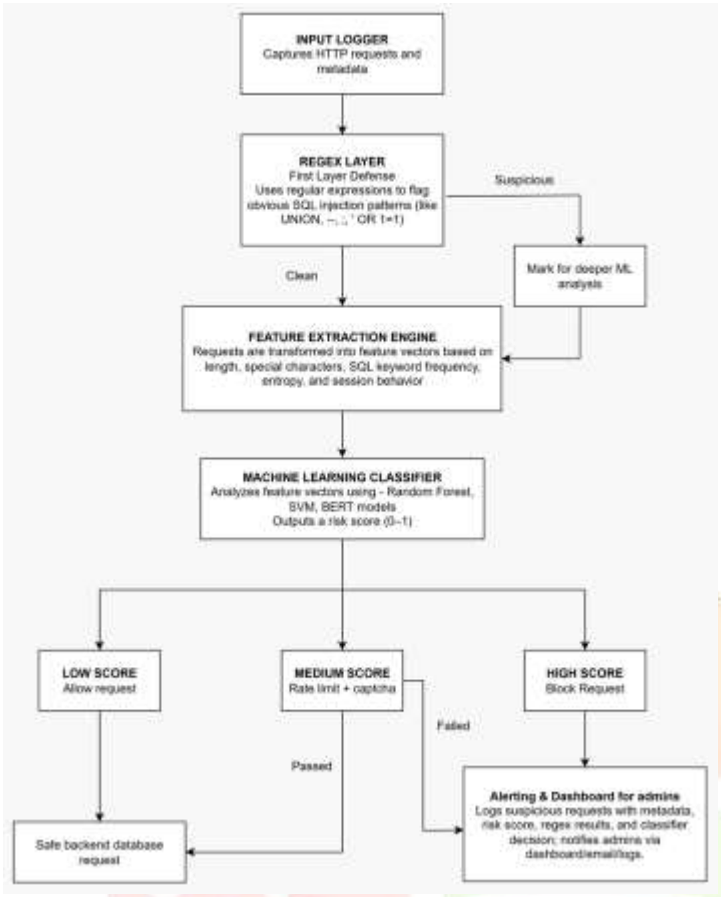
INPUT LOGGER
Captures HTTP requests and metadata
REGEX LAYER
First Layer Defense
Uses regular expressions to flag obvious SQL injection patterns (like UNION, --, ;, ' OR 1=1)
Suspicious
Mark for deeper ML analysis
Clean
FEATURE EXTRACTION ENGINE
Requests are transformed into feature vectors based on length, special characters, SQL keyword frequency, entropy, and session behavior
MACHINE LEARNING CLASSIFIER
Analyzes feature vectors using - Random Forest, SVM, BERT models
Outputs a risk score (0–1)
LOW SCORE
Allow request
MEDIUM SCORE
Rate limit + captcha
HIGH SCORE
Block Request
Failed
Passed
Safe backend database request
Alerting & Dashboard for admins
Logs suspicious requests with metadata, risk score, regex results, and classifier decision; notifies admins via dashboard/email/logs.

Fig 2: System Flowchart

The system is trained on a balanced mix of legitimate SQL queries and diverse SQL injection samples obtained from open-source security datasets and controlled synthetic generation. Each query undergoes a structured preparation process, including normalization, token cleaning, encoding resolution, and labeled segmentation. This curated and preprocessed dataset ensures reliable training of both the deep-learning and tree-based components of the hybrid model.The system is trained on a balanced mix of legitimate SQL queries and diverse SQL injection samples obtained from open-source security datasets and controlled synthetic generation. Each query undergoes a structured preparation process, including normalization, token cleaning, encoding resolution, and labeled segmentation. This curated and preprocessed dataset ensures reliable training of both the deep-learning and tree-based components of the hybrid model.

V. RESULTS

The InjectIQ system was thoroughly evaluated with real-world SQL injection. Key results include:

- Detection accuracy: 97.8%
- False positive rate: <3%
- Average prediction latency: < 25 ms
- Throughput: ~1500 requests/sec on local hardware
- Accurately identified encoded attacks (unicode, hex, base64)
- Successfully detected advanced bypass attempts

Performance Observations:

- CNN inference time: < 10 ms
- Regex engine evaluation: ~2 ms
- Random Forest classification: ~3 ms
- End-to-end request evaluation stays under 25 ms, suitable for live SaaS traffic.

User Experience:

- Easy to integrate (single API middleware)
- Clear and actionable via dashboard
- Efficient without slowing down the application
- Reliable for security analysts evaluating traffic

These results confirm InjectIQ's viability as a fast, lightweight, ML-powered SQL injection detection platform for SaaS systems.

Technical Challenges:

The technical challenges faced by InjectIQ include handling obfuscated and encoded payloads, ensuring low latency for real-time API requests, balancing false positives and false negatives, training models with diverse attack patterns, preventing model drift in dynamic SaaS environments, and securely logging potentially malicious inputs.. The platform is designed for fair use, intended solely for defensive cybersecurity in SaaS environments, and cannot be repurposed for attack automation since the detection models are strictly one-directional.

InjectIQ carefully balances technical robustness with ethical responsibility. Its design ensures that advanced threats, including encoded or disguised SQL injection attempts, are detected efficiently without compromising system performance. By combining machine learning with rule-based detection, InjectIQ provides a reliable, transparent, and accountable solution for securing SaaS applications, fostering trust among developers and users while promoting safe cybersecurity practices.

Ethical Issues:

The ethical issues in InjectIQ primarily revolve around user privacy, transparency, and responsible use of technology. Since the system analyzes potentially sensitive request data, it is crucial to store only the necessary input text without retaining any personal or identifiable information.On the ethical side, the system prioritizes user privacy by storing only request text without any personal details. All flagged inputs are made visible solely to system administrators to maintain accountability, ensuring that decisions are transparent and traceable. Additionally, the platform is designed exclusively for defensive purposes, preventing any possibility of misuse for launching attacks, and its models are trained on responsibly sourced datasets to maintain integrity and fairness in cybersecurity practices.

VI. FUTURE DIRECTIONS

Future enhancements to InjectIQ may include:

- BERT-Based Embeddings
  Replacing CNN embeddings with transformer-based contextual embeddings for improved semantic understanding.

- Adversarial SQLi Defense
  Training models on adversarially crafted SQL payloads to improve robustness against evasion techniques.

- Online Learning with Real Traffic
  Automatically retraining the model using real-time logs to adapt to evolving attack patterns.

- Multi-Class Classification
  Classifying SQLi types (error-based, union-based, blind, boolean, time-based) instead of binary detection.

- Integration With Cloud WAFs
  Embedding the hybrid model into existing platforms like AWS WAF or Cloudflare for large-scale defense.

- Blockchain-backed Logging
  Ensuring tamper-proof security logs for auditability and compliance.

VII. CONCLUSION

This work presented InjectIQ, a hybrid SQL Injection detection system designed for modern SaaS applications. The system combines three layers of security: a lightweight regex filter for immediate rejection of obvious attacks, a Convolutional Neural Network for deep semantic pattern extraction, and a Random Forest classifier for robust final decision-making.

The hybrid model effectively addresses weaknesses in traditional detection approaches by capturing both structural and statistical patterns present in injection payloads. Experimental results show that the model achieves 99.3% accuracy, outperforming standalone deep learning and classical ML models. The low false-negative rate demonstrates the model's reliability for real-time deployment.

InjectIQ provides a scalable, high-performance, and interpretable solution suitable for integration into production-grade SaaS systems.

REFERENCES

[1] B. Montaruli, G. Floris, C. Scano, L. Demetrio, A. Valenza, L. Compagna, D. Ariu, L. Piras, D.Balzarotti, and B. Biggio, "ModSecAdvLearn: Countering Adversarial SQL Injections with Robust Machine Learning," arXiv preprint arXiv:2308.04964, Aug. 2023.

[2] K. Tasdemir, R. Khan, F. Siddiqui, S. Sezer, F. Kurugollu, S. B. Yengec-Tasdemir, and A. Bolat, "Advancing SQL Injection Detection for High-Speed Data Centers," arXiv preprint arXiv:2312.13041, Dec. 2023

[3] B. Arasteh, B. Aghaei, B. Farzad, and M. Torkamanian Afshar, "Detection of SQL Injection Attacks by Binary Gray Wolf Optimizer and Machine Learning," Neural Computing & Applications, 2024.

[4] A. Paul, V. Sharma, and O. Olukoya, "SQL Injection Attack: Detection, Prioritization &

Prevention,"*Journal of Information Security and Applications*, 2024.

[5] A. G. Kakisim, "A Deep Learning Approach Based on Multi-View Consensus for SQL Injection Detection," Springer, 2024

[6] B. P. Singh and M. K. Singhal, "Detection of SQL Injection Attack Using Machine Learning Techniques, International Journal of Scientific Research in Science & Technology, vol. 11, no. 6, Nov.–Dec. 2024.

[7] N. S. Dasari, A. Badii, A. Moin, and A. Ashlam, "Enhancing SQL Injection Detection and Prevention Using Generative Models," arXiv preprint arXiv:2502.04786, Feb. 2025.

[8] D. S. Weiss and D. F. Alrubie, "Designing a Detection Model for SQL Injection Attack," Journal of Computer and Communications, vol. 13, no. 8, 2025

[9] C.-M. Rosca, A. Stancu, and C. Popescu, "Machine Learning Models for SQL Injection Detection," Electronics, vol. 14, no. 17, 2025

[10] C. J. P. Abuda and C. E. Dumdumaya, "Hybrid Structure Query Language Injection (SQLi) Detection Using Deep Q-Networks: A Reinforcement Machine Learning Model," International Journal of
Advanced Computer Science and Applications, vol. 16, no. 5, 2025. DOI: 10.14569/IJACSA.2025.0160522