© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712995

A Survey on Hybrid SQL Injection Detection: Feature-Selection, Classical Machine Learning, and Deep Learning Approaches to Obfuscated, Blind, and Time-Based SQLi

PUSHKAR Y JANE[1], ROSHANI K MUKADAM[2]
[1] *Department of Computer Science & Information Technology, Chhatrapati Shivaji Maharaj University, Panvel, Navi Mumbai*
[2] *Department of Computer Science & Information Technology, Chhatrapati Shivaji Maharaj University, Panvel, Navi Mumbai*

Abstract- ***SQL Injection (SQLi) remains one of the most persistent and damaging classes of web application vulnerabilities. As attackers adopt more sophisticated techniques — obfuscation, blind channels, and time-based inference — traditional detection techniques (rule/signature based and shallow ML) show limited robustness. Recently, hybrid approaches that combine feature selection, classical machine learning (ML) for fast filtering, and deep learning (DL) for semantic verification have gained traction. This survey thoroughly analyzes contemporary SQL Injection (SQLi) detection methods, specifically focusing on hybrid architectures capable of identifying obfuscated, blind, and time-based variants. This section details the SQLi attack taxonomy and corresponding defensive mechanisms. It further explores the application of advanced feature engineering techniques, such as Chi-Square ranking, to enhance detection. The analysis concludes with a performance evaluation (benchmarking) of both Machine Learning (ML) and Deep Learning (DL) models employed in this security domain.***

I. INTRODUCTION

SQL Injection (SQLi) constitutes a persistent, high-severity threat to web application confidentiality and integrity. The attack exploits input sanitization failures, enabling the injection of adversarial SQL fragments to illegally modify or bypass legitimate database operations. Current attacker methodologies necessitate detection systems capable of mitigating advanced evasion techniques, notably payload obfuscation, blind SQLi, and time-based data retrieval strategies. These variants are challenging because they often leave little or no overt evidence in typical logs, and they are engineered to bypass common filters. Detection techniques vary from Static Application Security Testing (SAST) and rule-based WAFs to data-centric ML/DL classifiers. Hybrid frameworks are emerging, utilizing multi-stage processing (e.g., feature selection, rapid ML screening, and final DL inspection) to achieve an optimal balance of throughput, precision, and security hardening. This survey synthesizes the latest findings and establishes the placement of these integrated, hybrid solutions in the broader research domain.

II. TAXONOMY OF SQL INJECTION ATTACKS

Before surveying defenses, we define a useful taxonomy of SQLi attacks:

1. Classic/Inline SQLi: Direct injection that returns data or raises errors, e.g., '; DROP TABLE users; --.
2. Obfuscated SQLi: Uses encoding (hex, Unicode), comments, whitespace, or token splitting to hide suspicious keywords, e.g., UN/**/ION SELECT.
3. Blind SQLi:
   - Boolean-based: The application returns similar pages but content varies depending on true/false condition.
   - Error-based blind: When the DB server reveals errors under some conditions.
4. Time-based SQLi: Uses deliberate delays (SLEEP, pg_sleep) to infer conditions by measuring response time.
5. Second-order SQLi: Payload stored and later executed in a different context.
6. Polymorphic/Automated SQLi: Payloads generated or mutated by tools/agents to avoid signatures.

Each variant imposes different detection requirements: obfuscation requires semantic understanding; blind/time-based attacks require behavioral and timing analysis.

© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712995

III. DETECTION TECHNIQUES — OVERVIEW

Detection approaches can be categorized as:

- Rule/Signature-Based Systems: WAF rules (e.g., ModSecurity). Fast but brittle to obfuscation and polymorphism.
- Static Source Code Analysis: Identifies insecure query construction patterns (concatenation of user input). Complements runtime detection but misses injected payloads at runtime.
- Dynamic/Runtime Anomaly Detection: Monitors live queries and responses for anomalies.
- Classical ML Methods: Feature vectors (lexical, syntactic, statistical) fed to classifiers (Logistic Regression, SVM, Random Forest, XGBoost).
- Deep Learning Methods: Sequence/structure models (LSTM, CNN, Transformers, CodeBERT) that learn token patterns or AST structures.
- Hybrid/Ensemble Approaches: Two- or multi-stage systems combining lightweight, fast detectors with heavier, more accurate models.

This paper concentrates on hybrid architectures that leverage feature selection and classical ML as a first stage and deep learning as a second, deeper stage.

IV. FEATURE ENGINEERING & FEATURE SELECTION

Feature engineering is crucial for ML/DL performance. Common feature categories:

- Lexical features: length, number of quotes/semicolons, special characters, ratio of non-alphanumeric chars.
- Keyword/Token frequency: counts of SELECT, UNION, OR, AND, SLEEP, --, etc.
- Syntactic/AST features: nesting depth, number of subqueries, parse trees.
- Entropy/statistical features: character distribution, compression ratio.
- Behavioral/timing features: response latency, size changes, HTTP status codes.
- Contextual features: request headers, user agent, previous session behavior.

Feature selection reduces dimensionality and noise. χ^2 (Chi-Square) ranking is often used to select features most correlated with the malicious/benign class. Studies report substantial gains in accuracy and reduced overfitting when χ^2 is applied prior to classical ML training. Alternatives include mutual information, recursive feature elimination (RFE), L1 regularization selection, and evolutionary algorithms.

V. CLASSICAL MACHINE LEARNING AS FAST FILTERS

Classical ML models are computationally efficient and often used as stage-one filters. Typical workflow:

1. Extract selected feature vectors (after χ^2).
2. Use fast classifiers: Logistic Regression, Random Forest, Naive Bayes, XGBoost.
3. Set asymmetric thresholds: aggressively mark obvious benign queries as safe; flag ambiguous/suspicious ones for deep analysis.

Advantages:

- Low latency, low resource usage (good for inline deployment).
- Interpretable: feature importances available for triage.

Limitations:

- Difficulty capturing semantic/structural obfuscation that changes tokens but not semantics.

Hence, hybrid systems delegate complex cases to deeper models.

VI. DEEP LEARNING FOR SEMANTIC VERIFICATION

Deep models provide the ability to model sequence and structure:

- Sequence models (LSTM/BiLSTM/GRU): learn token-level dependencies; useful for pattern recognition in token sequences.
- Attention and Transformer encoders: capture long-range dependencies; faster parallel training vs RNNs.
- Code-aware models (CodeBERT, Graph Neural Networks over ASTs): leverage structured representations and are well-suited to detect semantic obfuscation.
- Multi-modal models: combine token embeddings with behavioral/timing inputs.

Deep models are typically triggered only for suspicious queries (to limit latency). They excel at detecting obfuscation and polymorphic variants but

require careful training data and computational resources.

VII. DETECTION OF BLIND AND TIME-BASED SQLI

Blind and time-based attacks demand behavioral and timing analysis:

- For Boolean blind: compare responses to logically opposite probes and model subtle differences in HTML/text features (content similarity, DOM changes).
- For Time-based: measure response latency under controlled probes across multiple trials to detect statistically significant delays attributable to SLEEP or similar constructs.

Hybrid systems should incorporate timing features into both classical and deep stages. Sequence models can also learn patterns of queries that often accompany blind/time-based payloads (e.g., conditional constructs).

VIII. HYBRID PIPELINE DESIGN PATTERNS

Common hybrid patterns:

1. Filter → Verify: χ^2 + Random Forest filter reduces workload; suspicious queries passed to Transformer/LSTM.
2. Ensemble Consensus: Multiple independent classifiers (different features/architectures) vote; conflicting cases escalated.
3. Cascaded Confidence Thresholds: Confidence scoring used to decide if a query requires escalation.
4. Active Probing Module: For suspected blind/time attacks, system issues controlled probes and collects behavioral/timing signals for deeper analysis.

Design tradeoffs: latency vs accuracy, false positives vs negatives, throughput vs resource consumption.

IX. DATASETS AND EVALUATION PRACTICES

Public datasets are limited and often synthetic. Sources used in research include:

- Academic SQLi benchmark datasets (varied quality).
- OWASP labs: DVWA, WebGoat (for synthetic attack generation).
- Kaggle submissions and ad-hoc corpora.
- Custom datasets generated with tools (sqlmap) and LLM/Text-to-SQL generators.

Evaluation metrics: Precision, Recall, F1, ROC-AUC, False Positive Rate (FPR), detection latency, throughput. Robust evaluation must include obfuscation transformations, blind/time-based queries, and cross-domain generalization tests.

A persistent problem is a lack of large, diverse, publicly available datasets that include obfuscated and behavioral variants. Researchers often augment datasets with synthetic examples (template mutation, encoding transformations) and generative models (VAE/GAN) to broaden attack coverage.

X. COMPARATIVE ANALYSIS: STRENGTHS & WEAKNESSES

- Rule-based / WAFs: Simple, low latency, but brittle to obfuscation and polymorphism.
- Classical ML alone: Fast and interpretable, struggles with semantic obfuscation and blind/time channels.
- Deep Learning alone: High detection potential for obfuscation, but costly and risk of overfitting to limited datasets.
- Hybrid (Feature-Selection + ML + DL): Balanced — good latency via filtering, semantic understanding via DL, and lower overall resource use when appropriately staged.

Hybrid systems show the most promise for real-world deployments where throughput and robustness must be balanced.

XI. CHALLENGES & OPEN PROBLEMS

1. Dataset scarcity & realism: Need for large, labeled corpora covering obfuscation, blind, time-based, polymorphic, and code-generated SQL.
2. Adversarial adaptation: Attackers can adapt to ML/DL detectors; adversarial training and continual updating are required.
3. False positives: High FPR has operational costs — triage and human-in-the-loop mechanisms are necessary.

4. Latency and deployment: Ensuring DL verification does not introduce unacceptable delays in live systems.
5. Data privacy & log access: Access to production logs is sensitive — anonymization and privacy-preserving collection are required.
6. Explainability: Security teams need interpretable alerts to act; deep models tend to be opaque.
7. Testing for blind/time-based attacks in noisy networks: Distinguishing attack-induced delays from network jitter is nontrivial.
8. Security of automated pipelines: LLM/Text-to-SQL vulnerabilities and model backdoors can create new injection vectors.

XII. FUTURE RESEARCH DIRECTIONS

Promising directions include:

- Benchmark corpus creation: community efforts to publish large, annotated datasets with obfuscation and timing labels.
- Adversarially robust models: defenses against polymorphic payload evolution and adversarial examples.
- Lightweight code-aware transformers: optimized models (quantized/pruned) suitable for inline deployment.
- Active probing modules: safe probing strategies for blind/time detection with low false positive rates.
- Explainable hybrid systems: methods to provide human-readable explanations for DL decisions.
- Continuous learning pipelines: safe online retraining and concept-drift handling in production.
- Integration with secure code generation: vetting and hardening Text-to-SQL and AI code generation pipelines.
- Privacy-preserving telemetry: federated or differentially private learning across organizations.

XIII. CONCLUSION

Hybrid detection frameworks that combine feature selection, classical ML filters, and deep learning verification are well positioned to address contemporary SQLi threats — particularly obfuscated, blind, and time-based attacks. While the hybrid approach brings a pragmatic balance of speed and semantic power, success depends on rigorous dataset construction, adversarial robustness, deployment strategies that manage latency, and human-oriented explainability. This survey outlined the landscape, summarized current methods, highlighted gaps, and proposed actionable research directions that a PhD program can pursue to make meaningful advances.

REFERENCES

[1] C. Anley, “Advanced SQL Injection in SQL Server Applications,” NGSSoftware Insight Security Research, 2002.

[2] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[3] W. G. Halfond, J. Viegas, and A. Orso, “A classification of SQL-injection attacks and countermeasures,” in *Proc. IEEE Int. Symp. Secure Software Engineering*, 2006, pp. 13–15.

[4] W. G. Halfond and A. Orso, “Preventing SQL injection attacks using AMNESIA,” in *Proc. 28th Int. Conf. Software Engineering (ICSE)*, 2006, pp. 795–798.

[5] S. Bandhakavi, P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan, “CANDID: Preventing SQL injection attacks using dynamic candidate evaluations,” in *Proc. 14th ACM Conf. Computer and Communications Security (CCS)*, 2007, pp. 12–24.

[6] R. Valeur, D. Mutz, and G. Vigna, “A learning-based approach to the detection of SQL attacks,” in *Proc. Int. Conf. Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2005, pp. 123–140.

[7] J. Newsome, B. Karp, and D. Song, “Polygraph: Automatically generating signatures for polymorphic worms,” in *Proc. IEEE Symp. Security and Privacy*, 2005, pp. 226–241.

[8] A. Tajpour, M. J. Z. Ghahramani, and H. Ibrahim, “SQL injection detection using machine learning,” in *Proc. Int. Conf. Education and Management Technology*, 2010, pp. 40–43.

[9] R. Karimi, M. Fathy, and M. P. Fard, “Detection of SQL injection attacks using machine learning based techniques,” *Int. J. Computer Science and Information Security*, vol. 8, no. 6, pp. 1–6, 2010.

[10] J. Kim, J. Kim, and S. Kim, “SQL injection attack detection using character distribution and entropy analysis,” *IEEE Access*, vol. 7, pp. 162202–162214, 2019.

[11] Y. Zhang, Q. Zhang, and J. Yang, “Detecting SQL injection attacks using deep learning,” *Future Generation Computer Systems*, vol. 102, pp. 557–566, 2020.

[12] Z. Chen, J. Meng, and Y. Li, “Detecting SQL injection attacks based on convolutional neural networks,” *Applied Sciences*, vol. 10, no. 3, pp. 1–17, 2020.

[13] M. Roichman and R. Gudes, “Fine-grained access control to web databases,” in *Proc. ACM Conf. Computer and Communications Security (CCS)*, 2007, pp. 31–40.

[14] T. Mikolov et al., “Efficient estimation of word representations in vector space,” in *Proc. Int. Conf. Learning Representations (ICLR)*, 2013.