Volume 1 (2026), Issue 2
pp. 55–68

TSARKA SCIENCE

eISSN: 3134-6057
pISSN: 3134-6049

DOI: 10.66571/tsarka-3134-6057-07

REVIEW OF MACHINE LEARNING METHODS FOR PREVENTING FALSE-POSITIVE SQL INJECTION DETECTIONS

A.A. Askarov[1*], S.B. Mukhanov[1]

[1]Astana IT University, Astana, Kazakhstan,
*Corresponding author: datathink501@gmail.com.

Abstract

SQL injection remains a persistent web-application threat, yet practical detection systems often suffer from false positives that disrupt legitimate traffic and cause analyst fatigue. This review surveys machine learning methods for SQL injection detection with a focus on preventing false-positive outcomes in deployment, especially in Web Application Firewall (WAF) contexts. We define the scope across three observation layers—SQL query strings, HTTP request payloads/parameters, and derived network-flow features—and analyze how dataset design, feature engineering, and model selection influence false-positive behavior. We describe commonly used benchmark datasets and highlight limitations that hinder generalization, including synthetic traffic generation, label noise, multi-label ambiguity, and extreme class imbalance in web-attack traces. We compare classical machine learning models, deep learning approaches, hybrid systems, and anomaly-detection pipelines, emphasizing techniques that explicitly target false-positive reduction: threshold and risk-score tuning, probability calibration, cost-sensitive learning, ensemble strategies, adversarial training, and explainability for rule refinement. We summarize evaluation practices, recommending metrics suitable for imbalanced, low-base-rate settings and cost-aware decision-making. Finally, we discuss deployment constraints (latency, throughput, online learning, and concept drift) and articulate open research challenges and concrete recommendations for building reproducible, robust, low-false-positive SQL injection defenses.

Keywords: *SQL injection, machine learning, false positives, web application firewall, adversarial training, calibration, dataset realism, anomaly detection.*

1. Introduction

SQL injection (SQLi) is an injection-class vulnerability in which attacker-controlled input is incorporated into database queries, enabling unauthorized data access or modification and, in some cases, escalated operations [1]. SQLi is explicitly refer-

TSARKA SCIENCE
Volume 1 (2026), Issue 2
pp. 55–68
eISSN: 3134-6057
pISSN: 3134-6049

enced within the broader “Injection” risk category in the OWASP[2] Top 10 guidance, reflecting its continued relevance in web security practice [3].

The scope of this review is ML-based detection and prevention of SQLi with a narrow but operationally crucial target: reducing false positives (FPs). We treat “false-positive SQLi prevention” as preventing false-positive outcomes (unnecessary blocking/alerting) rather than preventing SQLi attacks themselves. This distinction matters because many ML papers optimize headline accuracy on benchmark datasets, while production WAF and intrusion-prevention scenarios require extremely low false-positive rates, robust operation under drift, and explainable decisions that can be tuned [4].

Definitions used throughout: - False positive (FP): benign activity incorrectly labeled as malicious/SQLi [4].

- False positive rate (FPR): FP / (FP + TN); when base rates are low, small FPR values can still create large absolute alert volumes [2].

- WAF context: a gateway that inspects HTTP requests and blocks or logs suspected attacks; FP handling is typically part of staged rollout, tuning, and threshold configuration [5].

SQLi types and why they matter for false positives. SQLi includes multiple exploitation styles that differ in surface form: for example, “blind SQL injection” infers database truth values via application response behavior rather than direct error output [6]. This diversity expands the “attack language” and increases overlap with benign strings (especially in user-generated content, search boxes, analytics parameters, or encoded payload fragments), which directly contributes to FP risk when detectors operate primarily on payload lexical cues [4].

False positives are intrinsic to security detectors and require tuning. The NIST[4] guidance for intrusion detection notes that false positives and false negatives cannot both be eliminated; reducing one often increases the other, and altering configurations to improve accuracy is “tuning.” [4] It additionally cautions that anomaly-based detectors often generate many false positives in diverse/dynamic environments and can be difficult to interpret for analysts—both issues directly relevant to ML-based WAF anomaly detection [4].

Thresholding is operationally central in WAFs. Many WAFs accumulate evidence from multiple rules and compare an overall score to a decision threshold; for example, documentation describing CRS-style anomaly scoring highlights that a total anomaly score is compared to a threshold to decide whether a request is malicious [5]. This reinforces why “FP prevention” is not only a classifier choice, but also a thresholding, calibration, and rollout problem.

Volume 1 (2026), Issue 2
pp. 55–68

TSARKA SCIENCE

eISSN: 3134-6057
pISSN: 3134-6049

2. Materials and methods

This review was conducted using a structured narrative methodology aimed at synthesizing primary, peer-reviewed sources concerning the application of machine learning to the detection of SQL injection (SQLi) attacks, with particular attention to the prevention of false-positive (FP) outcomes in operational deployments. The synthesis prioritized recency (publications from 2016 to 2026) and reproducibility, while seminal earlier works were retained where they define widely used benchmark datasets or foundational problem framings (e.g., HTTP CSIC 2010, ECML/PKDD 2007, and early anomaly-based WAF formulations) [7].

Search strategy. Bibliographic searches were conducted in IEEE Xplore, the ACM Digital Library, Scopus, ScienceDirect, SpringerLink, and Google Scholar, supplemented by targeted queries in arXiv to ensure coverage of recent open-access preprints. Search terms combined the following concepts and their lexical variants: “SQL injection detection,” “machine learning,” “web application firewall,” “false positives” / “false alarms,” “ModSecurity,” “OWASP CRS,” “adversarial,” “calibration,” “thresholding,” “concept drift,” together with established dataset names (CSIC 2010, CIC-IDS2017, ECML/PKDD 2007). The order of preference for source selection was: (i) peer-reviewed journal and conference publications; (ii) official dataset releases and reputable security research venues; and (iii) open-access preprints, where they represented the most accessible form of an influential recent contribution—a circumstance commonly observed in adversarial-machine-learning research targeting WAFs [8].

Inclusion and exclusion criteria. Studies were eligible for inclusion when they (i) addressed SQLi detection within HTTP, SQL, or query-context pipelines, or within WAF architectures; (ii) reported FP-reduction mechanisms applicable to SQLi detection systems, including probability calibration, precision–recall analysis, threshold tuning, or cost-sensitive learning [2]; or (iii) provided clearly described datasets and reproducible evaluation procedures. Sources lacking sufficient methodological detail to support analytical comparison, or whose evaluation setups could not be verified, were excluded or, where relevant for context, retained and labelled as limited-evidence.

Data extraction and analysis. From each included source, the following information was extracted where available: dataset and unit of observation (SQL query string, HTTP request payload, or network flow), feature engineering strategy, model family, evaluation metrics, FP-related results, and deployment context. Where datasets or metrics were not explicitly reported, this was recorded as “unspecified” rather than inferred. Extracted information was synthesized thematically along four axes: (i) dataset realism and label quality; (ii) feature engineering choices; (iii) model selection and FP-reduction techniques; and (iv) deployment-related considerations such as latency, online learning, and concept drift.

TSARKA SCIENCE

Volume 1 (2026), Issue 2
pp. 55–68

eISSN: 3134-6057
pISSN: 3134-6049

Limitations of the methodology. Reported metrics across the surveyed studies are not directly comparable because of substantial differences in dataset realism (synthetic versus production traffic), unit of analysis (payload, full HTTP request, or network flow), label definitions (binary versus multi-label), and train/test partitioning protocols. Dataset realism and label quality have been shown to materially influence performance estimates [9]; the comparative analysis presented in this review is therefore qualitative rather than quantitative.

3. Results and discussion

Timeline of influential artifacts (datasets, WAF-centric ML, adversarial work). (Conceptual timeline; not exhaustive.)

A conceptual timeline in the source review traces the development of FP-aware SQLi detection from early benchmark datasets such as ECML/PKDD 2007 and CSIC 2010 to recent WAF-centric learning, adversarially robust ModSecurity approaches, production audit-log datasets, and context-enriched request-response datasets.

Datasets used for SQLi detection research

Table entries emphasize what most affects false positives: realism, label definition, class balance, and whether the unit of analysis matches deployment (payload vs full request vs flow).

Table 1. Datasets used for SQLi detection research

Dataset / benchmark	Main use	FP relevance
1	2	3
ECML/PKDD 2007 [10]	Full HTTP requests; multi-label SQLi	Useful for context-rich FP analysis and label ambiguity.
HTTP CSIC 2010 [7]	Synthetic HTTP requests; normal vs anomalous	Strong baseline, but often optimistic for real traffic.
HttpParamsDataset [11]	Parameter payloads with explicit SQLi subset	Good for payload tests; weak endpoint context.
SR-BH 2020 [12]	Multi-label web-request dataset	Adds richer labels than simple attack/normal splits.
CIC-IDS2017 [13]	Network flows with web-attack SQLi	Severe SQLi imbalance; limited payload semantics.
LycoS-IDS2017 [14]	Corrected CIC-derived flows	Shows preprocessing errors can distort FP estimates.

Volume 1 (2026), Issue 2
pp. 55–68

TSARKA SCIENCE

eISSN: 3134-6057
pISSN: 3134-6049

1	2	3
ModSec-Learn [8]	Benign HTTP plus SQLi payloads for WAF learning	Closer to operational WAF decision boundaries.
WAF-A-MoLE [8]	Benign and malicious SQL queries for evasion tests	Useful for robustness studies; benign realism is limited.
Production ModSecurity logs [9]	30-day blocked-request corpus	High realism for alert-stream and rule-trigger analysis.
Request-response honeypot set [15]	Context-enriched labeled request-response pairs	Response context can reduce payload-only false positives.

Key dataset-driven FP insight: false positives are frequently data-defined. Synthetic datasets with limited endpoint diversity and sanitized structures can make benign traffic “too clean,” inflating separability and hiding the operational reality that benign inputs can contain attack-like substrings [9].

Feature engineering approaches

Feature design controls what “looks suspicious,” so it directly shapes FP behavior. The dominant families:

Table 2. Feature engineering approaches

Feature family	Main value	Main FP concern / note
1	2	3
Lexical tokens	Strong on classic signatures and obfuscation [10]	FPs rise when benign text contains SQL-like tokens [4].
Request structure	Uses field position and context to disambiguate input [10]	Usually lower FP than payload-only features.
String-shape statistics	Captures length, entropy, and encoding cues	Can flag long or encoded benign inputs [4].
Rule-trigger features	Turns CRS activations into operational ML signals [5]	Depends on rule quality, but learned weights can help [16].

TSARKA SCIENCE
Volume 1 (2026), Issue 2
pp. 55–68
eISSN: 3134-6057
pISSN: 3134-6049

1	2	3
Multi-label encodings	Support richer CAPEC-style labels [12]	Quality depends on how well encoding preserves meaning.
Network-flow features	Capture traffic-behavior patterns	Usually weak for content-level SQLi; FP instability is common [4].

ML models for SQLi detection

The model families below are grouped by what they optimize and how they tend to fail in FP-heavy settings.

Table 3. ML models for SQLi detection

Model family	FP-reduction strength	Main limitation
1	2	3
Classical supervised ML	Easy to calibrate and tune at low-FPR points [17]	May overfit lexical artifacts and drift [4].
Deep learning	Learns features automatically and can be calibrated	Needs strong benign data; FP debugging is harder [8].
Hybrid WAF + ML	Learns rule weights to improve TPR/FPR and latency [16]	Depends on rule coverage and service-specific data [16].
Adversarially robust WAF learning	Improves robustness while keeping false alarms low [18]	Needs continual threat-model updates [19].
One-class / anomaly detection	Useful when attack labels are scarce	Often high FP in dynamic environments [4].
Program repair / rule synthesis	Directly optimizes low-FP blocking [20]	Can be brittle and maintenance-heavy.

Techniques explicitly aimed at reducing false positives

False-positive prevention is best treated as a set of interventions spanning training, decision, and operations.

Volume 1 (2026), Issue 2
pp. 55–68

TSARKA SCIENCE

eISSN: 3134-6057
pISSN: 3134-6049

Thresholding and operating-point selection. Many security systems aggregate signals and compare them to a threshold; in CRS-style anomaly scoring, per-rule scores accumulate and are compared against a threshold [5]. In FP-sensitive deployments, results should be reported at the intended operating point (e.g., TPR at 1% FPR) rather than only as global accuracy [2].

Probability calibration. When models output probabilities/scores that guide blocking decisions, calibration can reduce "overconfident" mistakes that translate into FPs at low thresholds. Foundational work shows that methods such as Platt scaling and isotonic regression can significantly improve predicted probability quality [17].

Learning rule weights from data (hybrid rule+ML). ModSecurity/CRS pipelines can be reframed so that CRS rule activations are features and the system learns weights to improve the detection–FP trade-off, rather than relying on static heuristic severity weights; ModSec-Learn reports improved trade-offs and the possibility of discarding >30% of CRS rules via sparse regularization (which can reduce both FP and inference cost) [16].

Cost-sensitive learning and explicit FP penalties. In practice, FP costs are often higher than FN costs in revenue-critical endpoints (checkout, login, API calls). Operationally, this is reflected in the long-standing guidance that organizations tune systems by choosing a point on the FP/FN trade-off curve that fits their resources and risk posture [4]. Modern WAF-ML approaches operationalize this by optimizing detection at constrained low FPR (e.g., "negligible false alarm rates" while boosting detection) rather than maximizing unconstrained accuracy [18].

Ensembles and cascades. A common FP-reduction pattern is a cascade: low-cost coarse filtering (e.g., rule-trigger/keyword heuristics) followed by a high-precision model only on suspicious traffic. This also helps latency/throughput constraints, particularly when combined with caching/acceleration methods [21]. (Where specific ensemble details are not reported in a given primary source, they should be treated as design patterns rather than benchmarked results.)

Adversarial training and robustness evaluation. Adaptive attackers can manipulate SQLi strings to evade syntactic detectors without changing semantics. WAF-A-MoLE demonstrates this by applying semantic-preserving mutations and reporting that it bypasses the ML-based WAFs evaluated in that work [19]. ModSec-AdvLearn then uses adversarial training to improve both trade-off and robustness, reporting improvements such as increased detection (up to 30%) while retaining negligible false alarm rates, and robustness improvements (up to 85%) against adversarial SQLi manipulations [18].

Explainability for FP debugging and safe tuning. NIST highlights that anomaly-based alerts can be hard to interpret, complicating FP validation [4]. WAF rule-feature models (linear or sparse) provide an inherently actionable explanation channel: which

TSARKA SCIENCE
Volume 1 (2026), Issue 2
pp. 55–68
eISSN: 3134-6057
pISSN: 3134-6049

rules fired and what learned weights were applied, enabling targeted rule refinement and safer tuning than opaque end-to-end classifiers [16].

Evaluation metrics and reporting practices

Why many “high accuracy” results are insufficient. In imbalanced datasets and low-base-rate attack settings, accuracy and ROC curves can be misleading; precision–recall (PR) analysis is often more informative for imbalanced evaluation [2]. This is especially relevant to SQLi detection because realistic SQLi prevalence is typically far below 1% in benign traffic, while operational impact of FPs is immediate [4].

Recommended metric set (with FP emphasis): - Precision, recall, F1 (but interpret with base-rate awareness) [2].

- False positive rate (FPR) and true negative rate (TNR), since FP control is the dominant operational constraint in WAFs [4].

- PR curves / AUPRC for imbalanced settings [2].

- ROC / AUROC as secondary; AUROC can remain high even when precision collapses under imbalance [2].

- Cost-based metrics (expected cost per request or per alert), explicitly encoding FP vs FN cost ratios; NIST frames tuning as selecting the appropriate FP/FN balance for resources and risk [4].

- Robustness metrics under adversarial manipulation: performance on adversarial test sets, and degradation from clean to adversarial conditions [19].

Conceptual ROC vs PR guidance (qualitative):

ROC: TPR vs FPR (can look good under heavy imbalance).

PR: Precision vs Recall (directly reflects FP burden when positives are rare).

Key takeaway: when SQLi is rare, PR/AUPRC and precision-at-fixed-recall or recall-at-fixed-FPR are often more decision-relevant.

Deployment considerations: latency, throughput, online learning, and drift

Latency/throughput constraints. WAFs sit on the critical path. Optimization is therefore not optional: one WAF-focused system (RuleCache) reports performance gains via caching and online learning of rule ordering, with improvements ranging from single-digit to multi-fold depending on component and workload composition, and up to 5.5× total efficiency improvement in the reported prototype [21].

Online learning and tuning cycles. NIST explicitly describes tuning as configuration adjustment to balance FP and FN, and notes phased deployments to identify false positives early [4]. In WAF practice, this aligns with staged “detection-only” modes, shadow evaluation, and controlled threshold changes.

Concept drift and nonstationarity. Intrusion/web traffic is nonstationary; concept and feature drift are recurring concerns, and surveys emphasize that drift-aware IDS

Volume 1 (2026), Issue 2
pp. 55–68

TSARKA SCIENCE

eISSN: 3134-6057
pISSN: 3134-6049

methods remain underexplored relative to static benchmarking [22]. Drift is particularly relevant to FP control: evolving frameworks, new encodings, new user agents, and API changes can create benign distribution shifts that trigger anomaly detectors [22].

Data governance and privacy. Real-world HTTP request datasets must be anonymized; the 30-day production WAF dataset explicitly emphasizes anonymization while preserving structural tags like method, URI, triggered rule IDs, and headers [9]. This affects feature choices: models relying on raw content may be harder to share/reproduce, while rule-trigger features are easier to anonymize and standardize [9].

Representative performance and trade-offs (not directly comparable)

Because datasets and labeling differ substantially, the table below should be read as evidence of what classes of approaches can achieve, not as a leaderboard.

Table 4. Representative performance and trade-offs

Work / system	Key FP-related result	Main takeaway
1	2	3
Automated WAF repair [20]	Recall 54.6–98.3% at 0–2% FPR; tested against automatically generated bypass payloads on ModSecurity CRS rule sets across multiple application contexts.	Low-FP rule synthesis is practical.
ModSec-Learn [16]	Improves the detection–FP trade-off and prunes >30% of CRS rules using sparse (L1) regularization; evaluated on benign and SQLi HTTP requests with CRS rule activations as features.	Data-driven rule weighting can reduce FP and cost.
ModSec-AdvLearn [18]	Up to +30% detection at negligible false-alarm rates and up to +85% adversarial robustness; trained with semantics-preserving SQLi mutations on the ModSecurity feature space.	FP reduction should include adversarial robustness.

TSARKA SCIENCE
Volume 1 (2026), Issue 2
pp. 55–68
eISSN: 3134-6057
pISSN: 3134-6049

1	2	3
WAF-A-MoLE [19]	Successfully bypassed multiple ML-based WAF classifiers (e.g., Naive Bayes, Random Forest, SVM) using a guided-mutation engine that produces semantics-preserving adversarial SQLi payloads.	Precision tuning must be tested against evasion.
Production ModSecurity dataset [9]	Anonymized corpus of 147,205 HTTP requests blocked by OWASP ModSecurity over 30 days on a production web server; preserves HTTP method, URI, headers, and triggered CRS rule IDs.	Real traffic is essential for realistic FP analysis.
Injection-attack WAF ML eval [23]	AUROC for the SQLi class varies markedly across model families on web-attack benchmark data, with reported values ranging from approximately 0.7 to over 0.99 depending on classifier choice and feature representation.	Model choice and data mix strongly affect FP claims.
RuleCache [21]	Up to ~5.5× total throughput improvement on production-scale request workloads via online learning of CRS rule ordering and traffic-pattern caching.	Performance gains create room for higher-precision checks.

Practical synthesis: the strongest FP-focused results tend to (i) report performance at a fixed low FPR, (ii) incorporate domain constraints (WAF rules, context fields, request structure), and (iii) treat tuning/adaptation as part of the method rather than a postscript [16].

A practical FP-aware pipeline (conceptual)

HTTP request -> Parse and normalize (URL decode, canonicalize) -> Feature extrac-

Volume 1 (2026), Issue 2
pp. 55–68

TSARKA SCIENCE

eISSN: 3134-6057
pISSN: 3134-6049

tion.

Fast stage: CRS/rule triggers and lexical signals.

Low risk -> allow and log; uncertain -> high-precision stage with context-aware ML and calibration; high risk -> block/quarantine and log.

Analyst feedback is then used for rule and model tuning.

4.Conclusion

Summary of findings. This review has surveyed machine learning approaches to SQL injection detection through the lens of false-positive prevention in operational, particularly WAF-based, deployments. Three observations emerge consistently across the literature. First, false-positive behavior is, to a large extent, dataset-defined: synthetic and sanitized benchmarks tend to overstate separability, whereas datasets derived from production traffic and context-enriched honeypot pipelines reveal the structural ambiguity that drives FP rates in practice [9]. Second, FP reduction is most effectively treated as a multi-layer engineering objective rather than a property of a single classifier; threshold and risk-score tuning [5], probability calibration [17], cost-sensitive learning [4], hybrid rule-plus-model designs [16], and adversarial training [18] each address distinct failure modes and tend to be most effective in combination. Third, evaluation practices remain a key source of optimism bias: AUROC and global accuracy can mask precision collapse under low base rates, and only PR/AUPRC together with operating-point-specific metrics (e.g., TPR at 1% FPR) reliably reflect deployment-relevant performance [2].

Open challenges and future directions. Public evaluation remains constrained by dataset realism, labelling fidelity, and cross-domain generalization. Synthetic datasets such as CSIC 2010 remain useful but are structurally simpler than production traffic, while flow-level datasets (CIC-IDS2017 and its derivatives) frequently contain extremely small SQLi subclasses, rendering evaluation fragile and potentially misleading in the absence of PR- and cost-aware metrics [7]. Newer real-world WAF log corpora and context-enriched honeypot-generated request–response datasets are promising precisely because they better reflect operational ambiguity, which is the principal root cause of false positives [9]. Adversarial pressure can no longer be regarded as optional: mutation-based evasion work demonstrates that syntactic detectors—including ML-based ones—can be bypassed via semantics-preserving transformations, which motivates robust training procedures and continuous evaluation against an evolving threat model [19]. Finally, deployment realities—latency budgets, high throughput, and concept drift—must be incorporated at design time; acceleration approaches and drift-aware learning are key enablers of systems that combine low FP rates with high coverage [21].

Concrete recommendations. (1) False positives should be treated as an explicit optimization objective: performance should be reported at fixed low-FPR operat-

TSARKA SCIENCE
Volume 1 (2026), Issue 2
pp. 55–68
eISSN: 3134-6057
pISSN: 3134-6049

ing points (e.g., TPR at 1% FPR), and PR curves with AUPRC should be presented for imbalanced regimes [2]. (2) Evaluations should be designed around realistic benign distributions and endpoint diversity; real-world datasets should be used wherever possible, and the distinction between “blocked by the WAF” and ground-truth exploit success should be explicitly maintained [9]. (3) Probability calibration and cost-aware thresholding should be implemented in place of raw model scores when blocking decisions are made [17]. (4) Hybrid WAF-as-feature pipelines should be preferred for operational explainability and safer FP debugging; learned rule weighting and sparse selection have demonstrated strong potential for improving the detection–FP tradeoff while reducing inference cost [16]. (5) Adversarial evaluation, and where feasible adversarial training using problem-space transformations, should be incorporated into the development cycle, since real attackers actively optimize against the deployed decision boundary [19]. (6) Systems should be engineered for drift: FP rates should be monitored by endpoint, parameter, and client cohort, and phased rollouts and tuning cycles should be planned as part of the ML lifecycle [4]. (7) Latency budgets should be specified explicitly; when higher-precision analyses are introduced, acceleration and caching mechanisms should be employed to preserve throughput [21]. (8) Reproducibility should be improved by publishing datasets (or privacy-preserving surrogates), data splits, and decision thresholds, since FP behavior is, by its nature, threshold- and data-dependent [9].

Closing remarks. The principal contribution of this review is the consolidation of evidence indicating that effective machine learning–based SQL injection defense is not solely a question of model selection but a system-level engineering problem in which dataset realism, calibration, thresholding, adversarial robustness, explainability, and operational tuning must be co-designed. The most promising directions for future research therefore lie at the intersection of these dimensions: the development of context-rich, openly available datasets that reflect production conditions; the formalization of FP-aware evaluation protocols centered on PR analysis and operating-point metrics; and the construction of hybrid, drift-aware, adversarially robust pipelines that remain transparent to security analysts. Adoption of the recommendations outlined above is expected to support the development of SQL injection defenses that are not only accurate on benchmarks but also reliable, interpretable, and maintainable in real-world deployments.

References

1. OWASP Foundation. (n.d.). SQL Injection. OWASP Community. Retrieved from https://owasp.org/www-community/attacks/SQL_Injection

2. Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. PLoS ONE, 10(3), e0118432.

eISSN: 3134-6057
pISSN: 3134-6049

3. OWASP Foundation. (2021). A03: Injection. OWASP Top 10:2021. Retrieved from https://owasp.org/Top10/A03_2021-Injection/

4. Scarfone, K., & Mell, P. (2007). Guide to Intrusion Detection and Prevention Systems (IDPS). NIST Special Publication 800-94. Gaithersburg, MD: National Institute of Standards and Technology.

5. Microsoft. (2023). False positives elimination in Azure WAF. Microsoft Learn. Retrieved from https://learn.microsoft.com/en-us/azure/web-application-firewall/

6. OWASP Foundation. (n.d.). Blind SQL Injection. OWASP Community. Retrieved from https://owasp.org/www-community/attacks/Blind_SQL_Injection

7. Information Security Institute (CSIC). (2010). HTTP Dataset CSIC 2010. IMPACT Cyber Trust. Retrieved from https://www.tic.itefi.csic.es/dataset/

8. Floris, G., Scano, C., Montaruli, B., Demetrio, L., Valenza, A., Compagna, L., Ariu, D., Piras, L., Balzarotti, D., & Biggio, B. (2023). ModSec-AdvLearn: Countering Adversarial SQL Injections with Robust Machine Learning. arXiv preprint arXiv:2308.04964.

9. Lucz, G., & Forstner, B. (2025). A Thirty-Day Dataset of Malicious HTTP Requests Blocked by OWASP ModSecurity on a Production Web Server. Data, 10(11), 186.

10. Gallagher, B., & Eliassi-Rad, T. (2009). Classification of HTTP Attacks: A Study on the ECML/PKDD 2007 Discovery Challenge. Technical report. Lawrence Livermore National Laboratory.

11. Morzeux. (n.d.). HttpParamsDataset [Source code repository]. GitHub. Retrieved from https://github.com/Morzeux/HttpParamsDataset

12. Sureda Riera, T., Bermejo Higuera, J. R., Bermejo-Higuera, J., Martínez Herraiz, J.-J., & Sicilia, J. A. (2022). A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques. Computers & Security, 120, 102788.

13. LycoS-IDS Project. (n.d.). Intrusion Detection Evaluation Dataset (LYCOS-IDS2017). Retrieved from https://gitlab.com/cyber-iut/lycos-ids2017

14. Cantone, M., Marrocco, C., & Bria, A. (2024). On the Cross-Dataset Generalization of Machine Learning for Network Intrusion Detection. arXiv preprint arXiv:2402.10974.

15. Yu, H., Li, H., Shi, F., Yu, W., Ho, P., Wang, Z., & Wang, B. (2026). Multi-Agent Honeypot-Based Request-Response Context Dataset for Improved SQL Injection Detection Performance. arXiv preprint arXiv:2603.02963.

16. Scano, C., Floris, G., Montaruli, B., Demetrio, L., Valenza, A., Compagna, L., Ariu, D., Piras, L., Balzarotti, D., & Biggio, B. (2024). ModSec-Learn: Boosting ModSecurity with Machine Learning. arXiv preprint arXiv:2406.13547.

17. Niculescu-Mizil, A., & Caruana, R. (2005). Predicting Good Probabilities with Supervised Learning. In Proceedings of the 22nd International Conference on Machine Learning (ICML 2005) (pp. 625-632).

TSARKA SCIENCE

Volume 1 (2026), Issue 2
pp. 55–68

eISSN: 3134-6057
pISSN: 3134-6049

18. Floris, G., Scano, C., Montaruli, B., Demetrio, L., Valenza, A., Compagna, L., Ariu, D., Piras, L., Balzarotti, D., & Biggio, B. (2025). ModSec-AdvLearn: Countering Adversarial SQL Injections with Robust Machine Learning. IEEE Transactions on Information Forensics and Security.

19. Demetrio, L., Valenza, A., Costa, G., & Lagorio, G. (2020). WAF-A-MoLE: Evading Web Application Firewalls through Adversarial Machine Learning. arXiv preprint arXiv:2001.01952.

20. Appelt, D., Panichella, A., & Briand, L. (2017). Automatically Repairing Web Application Firewalls Based on Successful SQL Injection Attacks. In 28th IEEE International Symposium on Software Reliability Engineering (ISSRE) (pp. 339-350). IEEE.

21. Chen, X., Shen, Q., Cheng, P., Xiong, Y., & Wu, Z. (2022). RuleCache: Accelerating Web Application Firewalls by On-line Learning Traffic Patterns. In 2022 IEEE International Conference on Web Services (ICWS) (pp. 229-239).

22. Shyaa, M. A., Ibrahim, N. F., Zainol, Z., Abdullah, R., Anbar, M., & Alzubaidi, L. (2024). Evolving cybersecurity frontiers: A comprehensive survey on concept drift and feature dynamics aware machine and deep learning in intrusion detection systems. Engineering Applications of Artificial Intelligence, 137(Part A), 109143.

23. Durmuşkaya, M. E., & Bayraklı, S. (2025). Web application firewall based on machine learning models. PeerJ Computer Science, 11, e2975.

Information about authors

Askarov Anuar Askarovich – Master's student, Astana IT University, Astana, Kazakhstan
e-mail: datathink501@gmail.com
ORCID: https://orcid.org/0009-0009-2025-8838

Mukhanov Samat Bakytzhanovich – PhD, Astana IT University, Astana, Kazakhstan
e-mail: s.mukhanov@astanait.edu.kz
ORCID: https://orcid.org/0000-0001-8761-4272