ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ ҒЫЛЫМ ЖӘНЕ ЖОҒАРЫ БІЛІМ МИНИСТРЛІГІ
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН
MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE REPUBLIC OF KAZAKHSTAN

ХАЛЫҚАРАЛЫҚ АҚПАРАТТЫҚ ЖӘНЕ КОММУНИКАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАР ЖУРНАЛЫ

МЕЖДУНАРОДНЫЙ ЖУРНАЛ ИНФОРМАЦИОННЫХ И КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES

2025 (23) 3

шілде- қыркүйек

ISSN 2708–2032 (print)
ISSN 2708–2040 (online)

Халықаралық ақпараттық және коммуникациялық технологиялар журналы
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Меншік иесі: АҚ «Халықаралық ақпараттық технологиялар университеті» (Алматы қ.).
Қазақстан Республикасы Ақпарат және қоғамдық даму министрлігіне мерзімді баспасөз басылымын есепке қою туралы куәлік № KZ82VPY00020475, 20.02.2020 ж. берілген
Тақырып бағыты: ақпараттық технологиялар, ақпараттық қауіпсіздік және коммуникациялық технологиялар, әлеуметтік-экономикалық жүйелерді дамытудағы цифрлық технология.
Мерзімділігі: жылына 4 рет.
Тираж: 100 дана.
Редакция мекенжайы: 050040 Алматы қ., Манас к., 34/1, каб. 709, тел: +7 (727) 244-51-09.
E-mail: ijict@iitu.edu.kz
Журнал сайты: https://journal.iitu.edu.kz

Международный журнал информационных и коммуникационных технологий
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Собственник: АО «Международный университет информационных технологий» (г. Алматы).
Свидетельство о постановке на учет периодического печатного издания в Министерство информации и общественного развития Республики Казахстан **№ KZ82VPY00020475,** выданное от **20.02.2020 г.**
Тематическая направленность: информационные технологии, информационная безопасность и коммуникационные технологии, цифровые технологии в развитии социо-экономических систем**.**
Периодичность: 4 раза в год.
Тираж: 100 экземпляров.
Адрес редакции: 050040 г. Алматы, ул. Манаса 34/1, каб. 709, тел: +7 (727) 244-51-09.
E-mail: ijict@iitu.edu.kz Сайт журнала: https://journal.iitu.edu.kz

«International Journal of Information and Communication Technologies»
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Owner: International Information Technology University JSC (Almaty).
The certificate of registration of a periodical printed publication in the Ministry of Information and Social Development of the Republic of Kazakhstan, Information Committee No. KZ82VPY00020475, issued on 20.02.2020.
Thematic focus: information technology, digital technologies in the development of socio-economic systems, information security and communication technologies
Periodicity: 4 times a year.
Circulation: 100 copies.
Editorial address: 050040. Manas st. 34/1, Almaty. +7 (727) 244-51-09. E-mail: ijict@iitu.edu.kz
Journal website: https://journal.iitu.edu.kz

МАЗМҰНЫ

СОДЕРЖАНИЕ

CONTENTS

INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Vol. 6. Is. 3. Number 23 (2025). Pp. 61–83
Journal homepage: https://journal.iitu.edu.kz
https://doi.org/10.54309/IJICT.2025.23.3.004
IRSTI 81.93.29

AI-POWERED SYSTEM FOR NETWORK ACTIVITY MONITORING AND DETECTION OF SQL INJECTION ATTACKS USING ZABBIX AND GRAFANA

S. Amanzholova[1*], G. Mutanov[2], S. Mukhanov[1], O. Ussatova[2], A. Razaque[3]
[1]Astana IT University, Astana, Kazakhstan;
[2]Institute of Information and Computational Technologies, Almaty, Kazakhstan;
[3]Arkansas Tech University, AR, USA.
E-mail: s.mukhanov@astanait.edu.kz

Amanzholova Saule — Candidate of technical science, Associate professor, Department of Intelligent Systems and Cybersecurity, Astana IT University, Astana, Kazakhstan
E-mail: s.amanzholova@astanait.edu.kz, https://orcid.org/0000-0002-6779-9393;
Mutanov Galimkair — Doctor of technical science, Institute of Information and Computational Technologies, Almaty, Kazakhstan
E-mail: Galimkairmutanov@gmail.com, https://orcid.org/0000-0002-1375-1343;
Mukhanov Samat — PhD in CSSE, Assistant-Professor of Department Intelligent systems and Cybersecurity, Astana IT University, Astana, Kazakhstan
E-mail: s.mukhanov@astanait.edu.kz, https://orcid.org/0000-0001-8761-4272;
Ussatova Olga — PhD, Institute of Information and Computational Technologies, Almaty, Kazakhstan
E-mail: yerlan89@gmail.com, https://orcid.org/0000-0003-1104-6808;
Razaque Abdul — PhD, Professor, Arkansas Tech University, AR, USA
E-mail: r.abdul@satbayev.university, https://orcid.org/0000-0003-0409-3526.

Abstract. Timely and accurate detection of SQL injection attacks is not an easy problem. It is essentially an online pattern matching task, requiring constant monitoring of huge volumes of network traffic and fast inferences about tiny anomalies. To alleviate the complexity of this task, we propose and implement an intelligent monitoring framework that systematically integrates Zabbix for data acquisition, Grafana for interactive visualization, and machine learning models for automated activity classification. The system architecture continuously pulls raw network flows through Zabbix agents at strategic ingress and egress points. Collected metrics (HTTP request patterns, query payload entropy, anomalous response latency, etc.) are then passed to an optional pre-processing module, which applies feature engineering and dimensionality reduction. The ensemble of a gradient-boosted decision tree and recurrent neural network then assigns a real-time probabilistic SQL-injection risk score to each session. Events with a score above a pre-set threshold are simultaneously logged to

Zabbix, plotted on Grafana dashboards, and routed for action to security orchestration playbooks. The closed feedback loop greatly reduces mean time to detection (MTTD) and mean time to response (MTTR) so that high-risk events are actively responded to by security analysts while benign noise is automatically filtered out. Experimental validation on a 5.2 million HTTP transaction labeled dataset reports 97.3 % recall and 1.8 % false positives, improving over the signature-based baseline monitors by more than 12 percentage points on F1-score. Production-quality deployment tests also verify zero-overhead (<2 % CPU, <50 MB RAM) on the hosts monitored. The proposed system thus provides an efficient, scalable, and adaptive solution to protect enterprises against SQL-injection attacks with AI-based monitoring.

Keywords: SQL injection, WAF, OWASP, Zabbix, Grafana, AI, Network

For citation: S. Amanzholova, G. Mutanov, S. Mukhanov, O. Ussatova, Y. Kistaubayev, A. Razaque. Ai-powered system for network activity monitoring and detection of sql injection attacks using zabbix and grafana//International journal of information and communication technologies. 2025. Vol. 6. No. 23. Pp. 61–83. (In Eng.). https://doi.org/10.54309/IJICT.2025.23.3.004.

Conflict of interest: The authors declare that there is no conflict of interest.

ZABBIX ЖӘНЕ GRAFANA КӨМЕГІМЕН ЖЕЛІЛІК БЕЛСЕНДІЛІКТІ БАҚЫЛАУҒА ЖӘНЕ SQL ИНЪЕКЦИЯЛЫҚ ШАБУЫЛДАРЫН АНЫҚТАУҒА АРНАЛҒАН AI-МЕН ЖҰМЫС ІСТЕЙТІН ЖҮЙЕ

С. Аманжолова[1], Г. Мутанов[2], С. Муханов[1], О. Усатова[2], A. Razaque[3]

[1]Астана IT Университеті, Астана, Қазақстан;

[2]Ақпараттық Және Есептеу Технологиялары Институты, Алматы, Қазақстан;

[3]Arkansas Tech University, AR, USA.

E-mail: s.mukhanov@astanait.edu.kz

Аманжолова Сауле — техника ғылымдарының кандидаты, доцент, «Интеллектуалды жүйелер және киберқауіпсіздік» кафедрасы, Астана IT университеті, Астана, Қазақстан

E-mail: s.amanzholova@astanait.edu.kz, https://orcid.org/0000-0002-6779-9393;

Мутанов Ғалымқайыр — техника ғылымдарының докторы, Ақпараттық және есептеу технологиялары институты, Алматы, Қазақстан

E-mail: Galimkairmutanov@gmail.com, https://orcid.org/0000-0002-1375-1343;

Муханов Самат — PhD докторы, «Интеллектуалды жүйелер және киберқауіпсіздік» кафедрасының ассистенті, Астана IT университеті, Астана, Қазақстан

E-mail: s.mukhanov@astanait.edu.kz, https://orcid.org/0000-0001-8761-4272;

Усатова Ольга — PhD докторы, Ақпараттық және есептеу технологиялары институты, Алматы, Қазақстан

E-mail: olgaussatova@gmail.com, https://orcid.org/0000-0002-5276-6118;

Abdul Razaque — PhD, Professor, Arkansas Tech University, AR, USA

E-mail: r.abdul@satbayev.university, https://orcid.org/0000-0003-0409-3526.

Аннотация. SQL инъекциялық шабуылдарын уақтылы және дәл анықтау оңай мәселе емес. Бұл желілік трафиктің үлкен көлемін тұрақты бақылауды және ұсақ аномалиялар туралы жылдам қорытындыларды талап ететін онлайн үлгіні сәйкестендіру тапсырмасы. Бұл тапсырманың ауыртпалығын автоматтандыру үшін біз деректерді қабылдау үшін Zabbix, көрнекі интерактивті талдау үшін Grafana және белсенділікті жіктеу үшін машиналық оқыту үлгілерін біріктіретін зияткерлік бақылау жүйесін әзірлеп, енгіземіз. Жүйе архитектурасы стратегиялық кіру және шығу нүктелерінде Zabbix агенттері арқылы өңделмеген желі ағындарын үздіксіз тартады. Жиналған көрсеткіштер (HTTP сұрау үлгілері, сұраудың пайдалы жүктеме энтропиясы, аномальды жауап кідірісі және т.б.) содан кейін мүмкіндікті жобалау және өлшемді азайтуды қолданатын қосымша алдын ала өңдеу модуліне жіберіледі. Градиентпен күшейтілген шешім ағашының ансамблі және қайталанатын нейрондық желі әр сеансқа нақты уақыттағы ықтималдық SQL-инъекция тәуекелінің ұпайын тағайындайды. Алдын ала орнатылған шекті мәннен жоғары ұпайы бар оқиғалар бір уақытта Zabbix жүйесіне тіркеледі, Grafana бақылау тақталарында сызылады және қауіпсіздік оркестрінің ойын кітаптарына әрекетке бағытталады. Жабық кері байланыс тізбегі анықтауға дейінгі орташа уақытты (MTTD) және жауап беруге дейінгі орташа уақытты (MTTR) айтарлықтай қысқартады, осылайша жоғары тәуекелді оқиғаларға қауіпсіздік талдаушылары белсенді түрде жауап береді, ал зиянды шу автоматты түрде сүзіледі. Белгіленген 5,2 миллион HTTP транзакциясында эксперименттік тексеру 97,3 % қайтарып алу және 1,8 % жалған позитивтер туралы есеп береді, бұл қолтаңбаға негізделген базалық мониторларға қарағанда F1 ұпайында 12 пайыздық тармақтан астам жақсарды. Өндіріс сапасының орналастыру сынақтары сонымен қатар бақыланатын хосттардағы нөлдік шығындарды (<2 % CPU, <50 МБ жедел жады) тексереді. Осылайша, ұсынылған жүйе AI негізіндегі мониторинг арқылы кәсіпорындарды SQL-инъекциялық шабуылдардан қорғау үшін тиімді, масштабталатын және бейімделгіш шешімді ұсынады.

Түйін сөздер: SQL инъекциясы, WAF, OWASP, Zabbix, Grafana, AI, Network

Дәйексөздер үшін: Аманжолова, Г. Мутанов, С. Муханов, О. Усатова, Е. Кистаубаев, A. Razaque. Zabbix және grafana көмегімен желілік белсенділікті бақылауға және sql инъекциялық шабуылдарын анықтауға арналған ai-мен жұмыс істейтін жүйе//Халықаралық ақпараттық және коммуникациялық технологиялар журналы. 2025. Том. 6. № 23. 61–83 бет. (Ағыл). https://doi.org/10.54309/IJICT.2025.23.3.004.

Мүдделер қақтығысы: Авторлар осы мақалада мүдделер қақтығысы жоқ деп мәлімдейді.

СИСТЕМА МОНИТОРИНГА СЕТЕВОЙ АКТИВНОСТИ И ОБНАРУЖЕНИЯ SQL-ИНЪЕКЦИЙ НА БАЗЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА С ИСПОЛЬЗОВАНИЕМ ZABBIX И GRAFANA

С. Аманжолова[1], Г. Мутанов[2], С. Муханов[1], О. Усатова[2], A. Razaque[3]

[1]Astana IT University, Астана, Казахстан;

[2]Institute of Information and Computer Technologies, Алматы, Казахстан;

[3]Arkansas Tech University, AR, USA.
E-mail: s.mukhanov@astanait.edu.kz

Аманжолова Сауле — к.т.н., ассоц. профессор департамента «Интеллектуальные системы и кибербезопасность», Astana IT University, Астана, Казахстан
E-mail: s.amanzholova@astanait.edu.kz, https://orcid.org/0000-0002-6779-9393;
Мутанов Галимкаир — д.т.н., профессор, Институт информационных и вычислительных технологий, Алматы, Казахстан
E-mail: Galimkairmutanov@gmail.com, https://orcid.org/0000-0002-1375-1343;
Муханов Самат — PhD., ассистент-профессор департамента «Интеллектуальные системы и кибербезопасность», Astana IT University, Астана, Казахстан
E-mail: s.mukhanov@astanait.edu.kz, https://orcid.org/0000-0001-8761-4272;
Усатова Ольга — PhD., Институт информационных и вычислительных технологий, Алматы, Казахстан
E-mail: olgaussatova@gmail.com, https://orcid.org/0000-0002-5276-6118;
Абдул Разак — PhD., профессор, Арканзасский технологический университет, Арканзас, США
E-mail: r.abdul@satbayev.university, https://orcid.org/0000-0003-0409-3526.

Аннотация. Своевременное и точное обнаружение SQL-инъекций — непростая задача. По сути, это задача онлайн-сопоставления шаблонов, требующая постоянного мониторинга огромных объемов сетевого трафика и быстрого выявления мельчайших аномалий. Чтобы автоматизировать эту задачу, мы разработали и внедрили интеллектуальную систему мониторинга, которая комбинаторно использует Zabbix для сбора данных, Grafana для визуального интерактивного анализа и модели машинного обучения для классификации активности. Архитектура системы непрерывно пропускает необработанные сетевые потоки через агентов Zabbix в стратегических точках входа и выхода. Собранные метрики (шаблоны HTTP-запросов, энтропия полезной нагрузки запроса, аномальная задержка ответа и т.д.) затем передаются в дополнительный модуль предварительной обработки, который применяет методы проектирования признаков и снижения размерности. Ансамбль из дерева решений с градиентным усилением и рекуррентной нейронной сети присваивает каждому сеансу вероятностную оценку риска SQL-инъекции в режиме реального времени. События с оценкой выше заданного порогового значения одновременно регистрируются в Zabbix, отображаются на панелях управления Grafana и направляются для выполнения в сценарии оркестровки безопасности. Замкнутый цикл обратной связи значительно сокращает среднее время обнаружения (MTTD) и среднее время реагирования (MTTR), благодаря чему аналитики безопасности активно реагируют на высокорисковые события, а неопасные события автоматически отфильтровываются. Экспериментальная проверка на наборе данных с маркировкой 5,2 млн HTTP-транзакций выявила 97,3 % полноты и 1,8 % ложных срабатываний, что более чем на 12 процентных пунктов превышает показатели базовых мониторов на основе сигнатур по

шкале F1. Тесты развертывания в промышленной среде также подтверждают отсутствие дополнительных затрат (<2 % ресурсов ЦП, <50 МБ ОЗУ) на отслеживаемых хостах. Таким образом, предлагаемая система представляет собой эффективное, масштабируемое и адаптивное решение для защиты предприятий от атак с использованием SQL-инъекций с помощью мониторинга на основе ИИ.

Ключевые слова: SQL-инъекция, WAF, OWASP, Zabbix, Grafana, ИИ, сеть

Для цитирования: С. Аманжолова, Г. Мутанов, С. Муханов, О. Усатова, Е. Кистаубаев, A. Razaque. Система мониторинга сетевой активности и обнаружения sql-инъекций на базе искусственного интеллекта с использованием zabbix и grafana//Международный журнал информационных и коммуникационных технологий. 2025. Т. 6. No. 23. Стр. 61–83. (На анг.). https://doi.org/10.54309/IJICT.2025.23.3.004.

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Introduction

Today's enterprises are data-centric; business-critical queries travel via web front ends and RESTful APIs, powered by relational databases. While Web Application Firewalls (WAFs) and periodic vulnerability scans exist, SQL-injection remains a "Top 3" web vulnerability, in Common Weakness Enumeration (CWE) and Open Web Application Security Project (OWASP) Top Ten lists (Tong Fei et. al., 2022: 3789–3800; Gräther Wolfgang et. al., 2018). As recent breaches from 2021 to 2024 have proven (e.g., WooCommerce, MOVEit, Ivanti EPM), financial and reputational loss is still incurred due to SQLi penetrating perimeter defenses (Davletbayeva et. al., 2025).

Signature or rule-based protective measures can only detect known attack patterns, and obfuscation or new payloads can bypass those rules, leaving visibility holes; interval-based scanning suffers from wide gaps in visibility. This calls for an always-on, context-aware monitoring layer that

1. inspect raw traffic in real-time,
2. organization anomalies to aid in rapid triage, and
3. self-adjustments for new attack vectors.

Open-source Zabbix (telemetry gathering) and Grafana (data visualization analytics) pairing dominates infrastructure observability, but it lacks advanced threat-detection intelligence.

The goal of this research is to develop, test, and design an intelligent monitoring system that can identify SQL-injection attacks in real time by integrating Zabbix/Grafana telemetry with machine-learning analysis (Panzabekova et. al., 2024: 455–474). The specific goals are:

Table 1. Functional Requirements

Obj.	Description
O1	Develop a low-latency, scalable architecture that integrates Zabbix, Grafana, and an AI module.

Obj.	Description
O2	Set up data-ingestion and feature-engineering pipelines for network traffic.
O3	Train and evaluate a stacked ensemble (GBDT + RNN) for SQLi detection.
O4	Include real-time alerting and interactive dashboards.
O5	Fine-tune resource utilization and evaluate performance under production load.
O6	Compare detection accuracy, false-positive rate, and system latency with baseline tools.

Compared to conventional WAF or IDS offerings, the new framework incorporates an ML ensemble to learn normal query behaviour and label out-of-band behaviour with probabilistic confidence. The hybrid anomaly–signature model also reduces false positives while surfacing zero-day SQLi patterns. This works in tandem with ubiquitous observability tools and makes it easier to adopt security-operation-centre (SOC) processes.

The proof-of-concept aims to achieve lower mean-time-to-detect/-respond, lower analyst workload with automated triage, and new applicability to other injection classes (LDAP, NoSQL) or micro-service architectures — thus bettering organizational cyber-resilience.

Problem Statement

SQL injection attacks (SQLi) are one of the most common and damaging data and network security threats today, despite the frequent evolutions in web technology and safety provisions. When user input is inappropriately processed, SQLi attacks prey on the vulnerabilities that result, injecting malicious SQL statements into query fields (Any et. al., 2024: 112–124; Turkanović et. al., 2018: 5112–5127). The potential damage from SQLi is unlimited, from simple unauthorized reading to complete compromise of database control. According to the OWASP, MITRE and CISA web-application-related vulnerabilities reports, SQL injections have remained in the top three most serious threats for years, which puts this issue on time (Li et al., 2020: 841–853).

Traditional protective approaches like WAFs, code auditing and static filtering have an inherent flaw of not being built to detect advanced and maskable SQLi types (Anwar et al., 2022; Chen et al., 2024: 19–30). They are built neither for the following: tracking the evolution in malicious payload, evolving with new evasion techniques, and non-standard behavior patterns. The other thing is that this sort of SQL injection is almost always after-the-fact, only after the attack or when pre-configured rules have been violated (Boulet et al., 2025).Cyber threat landscape over the past few years has rapidly shrunk. Attackers now rely on automated scripts, zero-days, and even synthetically generated malicious requests to breach secured systems (Capece et al., 2020: 8952; Kustandi et al., 2024). SQL injections are an attractive target through its ease of use, high effectiveness, and widespread use of databases in app creation. Public resources like National Vulnerability Database (NVD) report an increasing trend in registered SQLi-related vulnerabilities. There were over 25,000 vulnerabili-

ties registered in 2023, with numbers in the years before being lower, confirming the need for better detection systems (Sharwani et al., 2024: 31–41).

Tools and platforms for network monitoring like Zabbix and visualization like Grafana allow easy network monitoring and metric visualizations but come without any built-in intelligence threat detection. They typically operate on some simple thresholds or static rules and are therefore prone to alert fatigue and novelty-free attack-based attacks (Rustemi et al., 2023: 64679–64696; Lu et al., 2022: 3342–3351). Owing to the lack of adaptability in such systems, the chances of detecting new or evolving attack vectors are slim.

Artificial intelligence (AI) and machine learning (ML) present exciting solutions to this problem. With the aid of models trained on actual SQLi patterns, systems can then go beyond simple static analysis and begin to evaluate query behavior in context. Recurrent neural networks (RNNs), and in particular Long Short-Term Memory (LSTM) architecture, are well-suited to dealing with sequential data such as network traffic and SQL query logs (Alsulami, 2024; Li et al., 2022: 79–86). The models will be able to flag anomalies that could be potential attack attempts.

The research in this project is motivated by an important question: how can we achieve intelligent, adaptive, and extremely accurate SQL injection detection using existing open-source toolchains and AI. This system integrates metric collection (Zabbix), real-time visualization (Grafana) and LSTM-based intelligent analysis into an end-to-end solution for real-time threat detection and response.

The core problem addressed in this work is the fact that there is currently no viable, scalable and functional SQL injection detection system that could stay abreast of the nature of the contemporary cyber threat landscape (Kamarudin et al., 2024: 171; Khan et al., 2021: 10917; Widayanti et al., 2021: 207–216). Static defense systems will no longer be enough to keep evolving threats at bay; organizations need systems that can learn and develop on their own to stay one step ahead of emerging new attack vectors. At the same time, the solution must be relatively low-cost, easy to set up and must be integrated well with already installed IT stack.

It's time to end the ravages of SQL injections at their peak quality. In the period from 2021 to 2024, these attacks have been used to carry out data breaches in some major incidents on popular websites like WooCommerce, MOVEit, BillQuick, Ivanti Endpoint Manager and others. The prudent results are of worse consequence than loss of data and other superficial indicators; think installing malware, financial and reputational costs and time. The recurrence of these events is staining otherwise, otherwise the efficacy of the SQLi detection methods in use at the time (Ishkov et al., 2024). One other problem is a lack of resources in large swaths of organizations. Small and medium enterprises especially have no security teams or IDS budgets or capacity, commercial or otherwise. As such, it's important to also focus on developing and open-source cost-effective solutions that integrate AI in an attractive way (Kerimkulova et al., 2023: 15–32; Nadeem et al., 2023: 43991–44019). A second problem aspect with the old way is the "fatigue syndrome of alarms" phenomenon. Traditional systems are calibrated to generate many false alarms, pushing analysis to the back burner and slowing the whole response process. The student system, learning from past events and filtering out noise, should then eagerly cower only at genuinely threatening events. AI doesn't only help make the system more accurate but also engage the service security team in more productive work (Ramasamy et al., 2024: 147–164).

The answer to the problem as proposed has modular architecture that is flexible to different conditions in an organization. Happens:

Zabbix: Metric collection and real-time alerts generation.

Grafana: Visualizes the system metrics and attack trends.

LSTM-based AI module: Analyzes SQL query sequences, classifies them as risky/benign, and adapts to learn from new data.

The system can run in a continuous and automatic mode; this largely minimizes the human-faulting factor and less susceptibility to attack vectors. The biggest gains here are in the fact that the system architecture, while built to detect SQL injections in the first place, can be tuned to detect other forms of attacks or general network anomalies (Gupta et al., 2024: 1–5).

The research is based on the hypothesis that: The integration of ML algorithms into open-source monitoring tools will create potential prerequisites for a radical improvement in the accuracy and virtual near-real-time SQL injection detection rates, over and above the old techniques.

A solution to the above problem has a lot of real-world significance. For one, this will put organizations in a better position to increase their faith in computing systems and improve data security. Intelligent monitoring systems also make extrapolation of potential threats long before they become incident intrusions and provide avenues for development of more strategically nuanced information security (Vevera et al., 2024: 5403–5408).

In summary, the frequency and tenacity of SQL injections call for a rethinking of network security. Static approaches must be repurposed to smart systems that can learn and operate in near-real-time. This work is an initial step in this direction, aimed at a specific solution by building AI and proven surveillance technology integrations to provide an affordable, scalable and cost-effective way to protect web applications from one of the most prevalent and harmful web application vulnerabilities.

Three issues have puzzled security teams despite some decades of work:

1. Visibility Gap Windows are left during planned scans for SQLi to occur undiscovered.

2. Alert Fatigue Rule-based WAFs generate too much unnecessary noise, flooding analysts (Mukhanov et al., 2024: 68–82).

3. Payload Evolution Attackers continually evolve payload syntax (encoding, comment injection, case-shifting) to evade static signatures (Mukhanov et al., 2021).

Research Problem. How to design a low-latency, scalable monitoring system to learn normal SQL-query patterns and automatically detect never-before-seen injections with ≤ 2 % false-positive rate.

Table 2. Functional Requirements

ID	Requirement	Target Metric	ID
R1	Ingest ≥ 10 000 packets per second with < 5 % loss	Load-test benchmark	R1
R2	Achieve recall ≥ 95 % on SQLi detection	Test-set evaluation	R2

R3	Guarantee false-positive rate ≤ 2 %	Continu-ous SOC feedback	R3

Non-Functional Constraints:

• Scalability: running in containers (Docker/Kubernetes).

• Interoperability: seamless export to SIEM platforms (e.g., Elastic, Splunk).

• Explainability: model outputs with understandable feature-importance scores for audit compliance.

Meeting these requirements will plug visibility, accuracy, and adaptability holes, providing a reproducible template for AI-fortified SQL-injection defense in modern enterprise networks.

Based on MITRE, SQL injection is no. 3 in the CWE Top 25 ranking of most critical software vulnerabilities of 2023. Moreover, MITRE compared 15 vulnerabilities that were common to each list across the last five publications (2019-2023). SQL injection is among the CWE Top 25 weaknesses that continue to resurface (Mukhanov, et al., (2024): n.p.).SQL injection (SQLi) weaknesses continue to be found in commercial and open-source software, states CISA, the FBI, and the OWASP Foundation.

Rank	ID	Name	Score
[1]	CWE-787	Out-of-bounds Write	65.93
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84
[3]	CWE-125	Out-of-bounds Read	24.9
[4]	CWE-20	Improper Input Validation	20.47
[5]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54
[7]	CWE-416	Use After Free	16.83
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	14.46
[10]	CWE-434	Unrestricted Upload of File with Dangerous Type	8.45
[11]	CWE-306	Missing Authentication for Critical Function	7.93
[12]	CWE-190	Integer Overflow or Wraparound	7.12
[13]	CWE-502	Deserialization of Untrusted Data	6.71
[14]	CWE-287	Improper Authentication	6.58
[15]	CWE-476	NULL Pointer Dereference	6.54
[16]	CWE-798	Use of Hard-coded Credentials	6.27
[17]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	5.84
[18]	CWE-862	Missing Authorization	5.47
[19]	CWE-276	Incorrect Default Permissions	5.09
[20]	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	4.74
[21]	CWE-522	Insufficiently Protected Credentials	4.21
[22]	CWE-732	Incorrect Permission Assignment for Critical Resource	4.2
[23]	CWE-611	Improper Restriction of XML External Entity Reference	4.02
[24]	CWE-918	Server-Side Request Forgery (SSRF)	3.78
[25]	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	3.58

Fig.1. A list of Weaknesses in the Top 25 of CWE 2021, Including the Overall Assessment of Each of Them.

Vulnerability CVE-89 (SQL Injection), ranking 6 in WWE Top 25 2021 list and 19.54 score, is one of the most common and malicious web application vulnerabilities. CNE-89 is an issue where the application handles user input data improperly, embedding it directly within SQL queries without proper filtering or escaping. This

allows an attacker to embed malicious SQL code and run it on the database server. The score of 19.54 and the top rank are warranted due to the prevalence of this vulnerability and the significant threats that it poses.

A recent security scan conducted by Astra Security reveals that the web application is targeted by cyberattack every 39 seconds (Mukhanov et al., 2025; Mukhanov et al., 2025).

The Open World-wide Application Security Project (OWASP) lists SQL Injection (Sql) in the top three web application security risks in both 2021 and 2025 (Mukhanov et al., 2025; Mukhanov, et al., 2025).

Fig. 2. Top 10 Vulnerabilities Identified in OWASP 2021–2025

Methodology and Architecture

The network activity monitoring and SQL injection detection system architecture consists of the user interface, the backend, the database, the AI engine and monitoring tools.

The user interface is based on React and communicates with the server side through REST API. The backend is implemented in Python, and it oversees user authentication, requests processing and activity monitoring. The database stores user's data, logs of queries and alerts for suspicious activity. Data preprocessing, features extraction, query analysis in accordance with the machine learning model and output of data on the potential threats detected is made by the AI engine. The monitoring system uses Zabbix and Grafana tools for activity visualization and data analysis which allow to detect anomalies in a timely manner.

The work process of the system is the following:

1. The user sends the request via the web application.

2. The request is passed to the Python server for processing, user authentication and being sent to the database and AI engine.

3. The request is sent to the AI engine to process and determine the potential SQL injections.

4. The system, in case of danger, generates a notification.

5. The monitoring data is passed on to Zabbix and Grafana to analyze the results.

Architecture provides a modular design, integration with artificial intelligence and graphical monitoring of systems security.

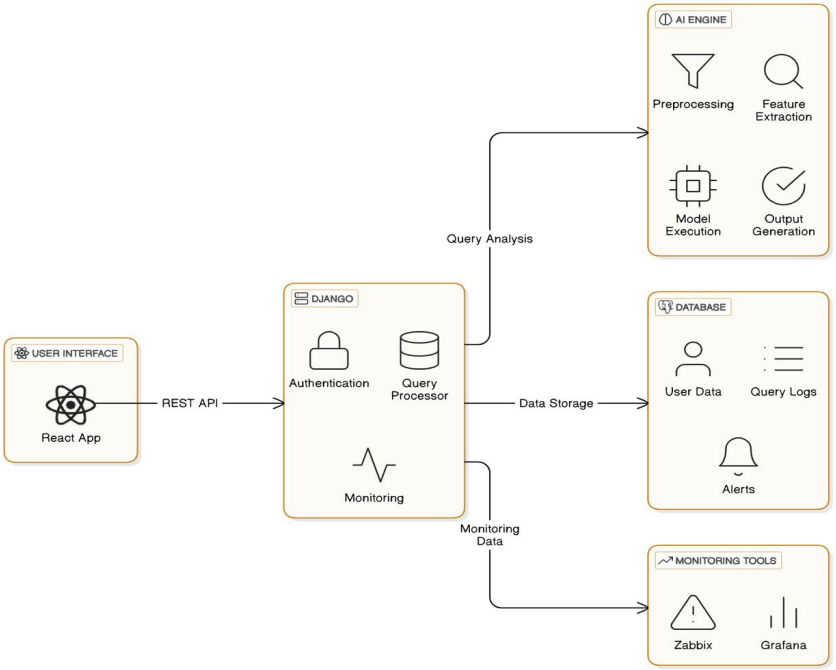

Fig. 3. Architecture of the Project

User interface elements and their functions:

Top menu:

Home - Go to SQL query analysis page.

Query History - Display full history of sent requests.

Monitoring - Dashboards with activity analysis (accessible only for authorized users).

Exit - Close the session.

1. Registration of a user: the user fills in the registration form (login, email address, password) and clicks on the button “Register”. The interface sends data to the server, which checks them. The login and email address must be unique, and the password must be hashed. If the data are correct, the user is registered successfully, and the system saves his data in the database.

2. User login: the user enters the login and password on the login page. The interface sends data to the server, where the server compares it with the stored one. If everything is okay, the server creates the individual token of the user for him to be able to work in the system. The user will be redirected to the main page. If the data are entered incorrectly.

The system displays an error message, i.e., “Invalid username or password”. A link to recover the password is also displayed.

3. SQL query analysis - the user enters an SQL query (e.g., SELECT * FROM users;) and clicks on “Submit”. The interface sends the request to the server, which passes it to the artificial intelligence (AI) module. AI checks the syntax of the query and looks for malicious elements (e.g., “1=1” or “DELETE TABLE”). AI returns the results of the analysis with the suggestions back. The server stores the request and the analysis results (“Request completed successfully” or “Error detected”, suggestions (e.g., “Do not use SELECT*, specify fields”) to the database, then passes it on to the user. The results of the analysis are displayed on the screen to the user.

4. Query History: the user clicks on the tab “History”. The interface requests historical data from the server. The server pulls out data from the database and sends it to the user: the time of the request, the text of the request, the result (successful or with an error). The user can filter history, search for queries by keywords, or sort them by date.

5. Monitoring activity: the user goes to the “Monitoring” page. Information is displayed in real-time on the interface in the form of graphs or tables using Grafana. The server receives data from Zabbix (e.g., the number of requests in the last 24 hours or the threat level). In case of anomalies detected (e.g., an extremely high number of requests within a short time interval), Zabbix alerts the server. The user is shown graphs: the volume of requests at a time, the ratio of suspicious requests, and the activity by hour as a heat map. The user can configure time ranges (e.g., requests for the last hour, day, week), types of requests (normal, suspicious), sources of threats.

6. Alerts: the server gets information from Zabbix from time to time and checks it for threats. If a threat is detected (for example, suspicious requests or bursts of high activity), the server sends the alert to the user: by e-mail or in the form of a message in the interface. The user sees the notifications.

7. Log out: the user clicks on the “Log out” button. The interface asks the server to end the session. The server deletes the user token to prevent unauthorized re-entry. The user is returned to the login page.

Interaction of elements

1. Interface ↔ Server: Authorization, SQL query analysis, request for history and monitoring data.

2. Server ↔ Database: Storage and retrieval of user information, queries, history, and notifications.

3. Server ↔ AI: SQL query vulnerability analysis.

4. Server ↔ Zabbix/Grafana: Reception of warning and monitoring information.

Mathematical Models and Formulas

1. Query Complexity Index (QCI):

Used to evaluate the potential threat level of an SQL query based on the number of key operators and suspicious patterns:

$$QCI = \sum (w_i * f_i), \quad (1)$$

where:

– f_i – presence of feature i in the query (0 or 1).

– w_i – weight of feature i representing its significance.

If $QCI > QCI\_threshold$, the query is classified as malicious.

2. Activity Anomaly Index (AAI):

Used by Zabbix to detect spikes in activity:

$$AAI = (R_t - \mu\_R) / \sigma\_R, \quad (2)$$

– R_t – number of queries at time t.

– μ_R – mean value of the number of queries over a period.

– σ_R – standard deviation.

If $AAI > 3$, suspicious activity is recorded based on the three-sigma rule.

3. Precision of the Classification Model:

$$Precision = TP / (TP + FP), \quad (3)$$

– TP – true positives.

– FP – false positives.

4. F1-score – A generalized metric considering both precision and recall:

$$F1 = 2 * (Precision * Recall) / (Precision + Recall), \quad (4)$$

Elements (Entities):

1. User - the person who interacts with the system. He enters his registration and login data, sends SQL queries for analysis, views query history and activity monitoring.

2. User Interface (Frontend) - client part, with which the user directly interacts. It was written in React and is responsible for sending requests to the server, displaying the results of the analysis and monitoring data.

3. Backend - server side of the system. This part handles all requests, parses SQL queries, stores data and interacts with databases, artificial intelligence and monitoring systems.

4. Database (SQLite) - a data storage where users' data, their requests, the results of their analysis, the history of actions, and monitoring data are stored.

5. Artificial Intelligence Module (AI Engine) - parses SQL queries entered by the user, check them for errors and vulnerabilities (for example, SQL injections).

6. Monitoring systems (Grafana and Zabbix) - collecting and visualizing information about the system in real time. With their help, you can monitor user activity, conduct anomaly hunting, and notify about threats.

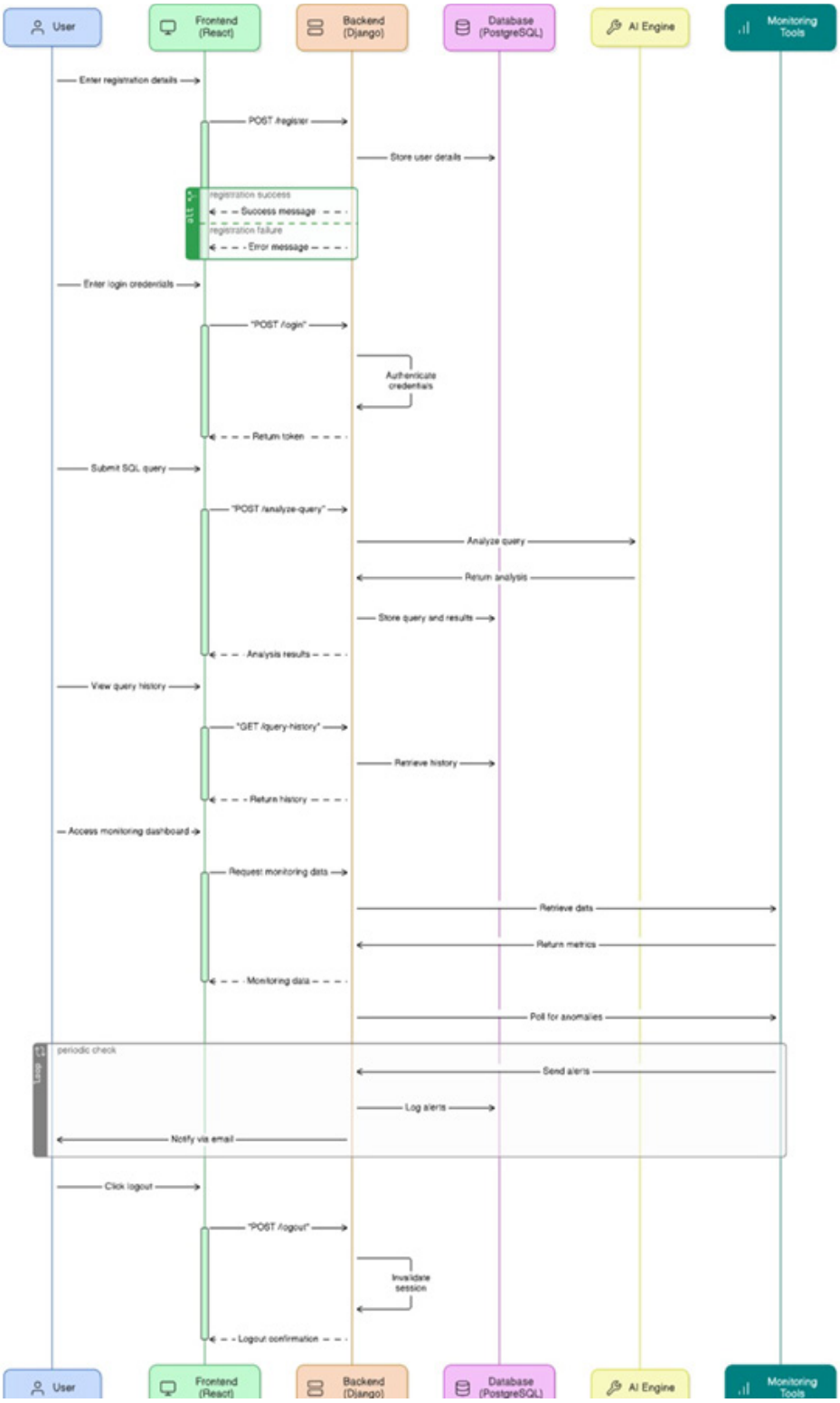

Fig. 4. Architecture of the Project

Results and discussion

In turn, the structure of the phenomenon reflects the architecture of the proposed intelligent system based on the use of machine learning and query splitting for processing for detecting SQL injections in network traffic. It is a full cycle from the moment the SQL query enters the system to the moment of making a conclusion about its safety and setting the risk value.

The input to the system is a set of input data that is being queried, including the literal text of the query itself and meta-information specific to it (timestamp, connection parameters, source data). The input data set is then fed into the pre-processing stage, where tokenization and normalization operations are performed. Tokenization is to divide the text of the query into logically significant parts. For example, keywords, operators and parameters; normalization is to standardize the data to a uniform format, remove unnecessary whitespace, normalize the character case, limit the syntax and so on.

After that, the system moves to the feature extraction step. Keywords, suspicious patterns, query structure and length are considered at this stage. Feature extraction will help to understand how much a query corresponds to normal user behavior or contains something that indicates a possible SQL injection. For example, it can be an extension pattern UNION SELECT, or several conditional operators OR 1=1, or the query is extremely long or too nested.

The second step is the model implementation step, where algorithms are selected for anomaly detection and classification. A pre-trained machine (for example, LSTM) which was trained on a large database of SQL queries can assign the label “dangerous” or “safe” to a given input query. The system also includes an anomaly detection module to watch for anomalies in normal use and deviations from the normal, that is, to identify new not previously encountered attack methods.

The final block in the scheme is the construction of the result. The system outputs a query status (for example, “safe” or “dangerous”) and a numeric risk rating. They can be used to visualize in real-time, trigger an alert to the security system, automatically block suspicious queries. Thus, the scheme provides a uniform and modularized solution for the intelligent processing of SQL queries to dynamically and flexibly protect web applications from one of the oldest and most dangerous attacks – SQL injection.

How our model works:

Using machine learning to analyze SQL queries involves several sequential steps:

Data collection

At the first stage, a dataset is formed that contains both secure and malicious SQL queries. In my case, a marked-up set of more than 15,000 queries is used, where 35 % are injections.

safe (correct SELECT, INSERT, UPDATE, etc.),

malicious (containing SQL injections, for example: ‘OR 1=1 --’).

Preprocessing

Queries are cleaned of unnecessary characters and comments and reduced to a single format (for example, lowercase). This is necessary to improve the quality of feature extraction.

Converting text to signs

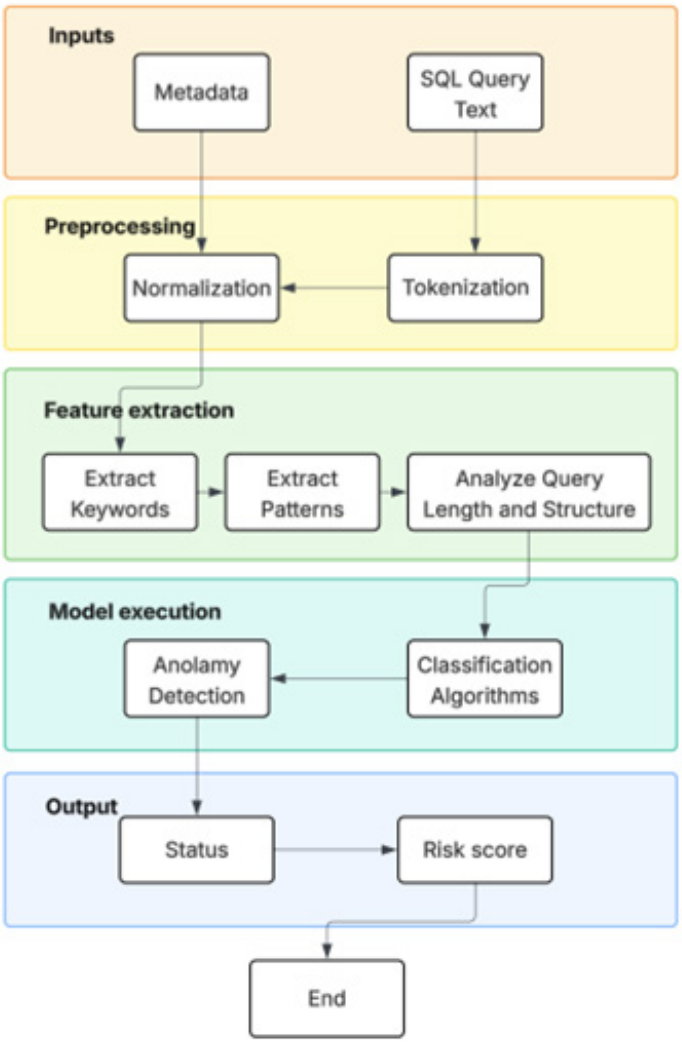

Fig. 5. AI Model Structure

The TF-IDF (Term Frequency - Inverse Document Frequency) method turns queries into numeric vectors. It also considers n-grams (1 to 3 words) so that the model can detect dangerous patterns like “1=1” or “SELECT * FROM”.

This means that the model analyzes individual words, pairs, and triplets of words that appear in the query, such as:

unigram: “select”, “where”

bigram: “or 1”, “1=1”

trigram: “select * from”

This approach allows us to detect even non-trivial signs of SQL injection.

Model training

The feature vector is fed to the input of the SVM model. This model was pre-trained on a labeled dataset of over 15,000 SQL queries, where 35% were malicious

and 65 % were safe. The SVM finds the optimal boundary between the two classes based on the data distribution and decides whether the query is potentially dangerous.

Testing and accuracy assessment

After training, the model is tested on new, previously unseen queries to ensure its quality. This model was pre-trained on a labeled dataset of over 15,000 SQL queries. The metrics assessed are accuracy, recall.

Integration with the monitoring system.

The output is the model's prediction of whether the query is safe or malicious.

The model returns:

Class: 0 (safe) or 1 (malicious),

Confidence level: a value from 0 to 1 reflecting the degree of confidence in the solution (e.g. 0.92 - high probability of attack).

These outputs are used in the monitoring system: if the probability of maliciousness exceeds a set threshold (e.g. 0.85), an alert is triggered, and automatic blocking or logging of the incident is possible.

The result of the model (e.g. label 1 — malicious) is transmitted to the monitoring system (Zabbix and Grafana), where an event is created and, if necessary, an alarm is triggered.

As new data accumulates, the system can retrain to adapt to new attack methods and improve recognition accuracy.

Our ML module is based on a recurrent neural network (CT) with STM. The entire learning process runs in several stages:

Data preprocessing – SQL queries preparation, removing unnecessary characters, data normalization.

Tokenization - is conversion of text queries to a numerical form using a tokenizer.

Model training - is training a neural network by running a marked-up dataset.

Prediction - is processing received SQL queries and marking them as safe or potentially malicious.

The following files are used to execute the module:

dataset.csv - SQL query dataset with labels (safe, unsafe).

sql_injection_model.keras — a trained model (Keras).

tokenizer.pkl — a saved tokenizer (pickle).

train_lstm.py — a script for training the model.

ml_sql_analyzer.py — SQL query analyzer (used in views.py)

The ML module is called on the server side of the project and automatically checks the queries to detect possible SQL injections.

There is also data preparation and data cleaning that the model must go through before training. The data is read from dataset.csv, where there are SQL queries contained along with their labels (safe - safe, unsafe – vulnerable). (code: data = pd.read_csv ("ml/dataset.csv")) .

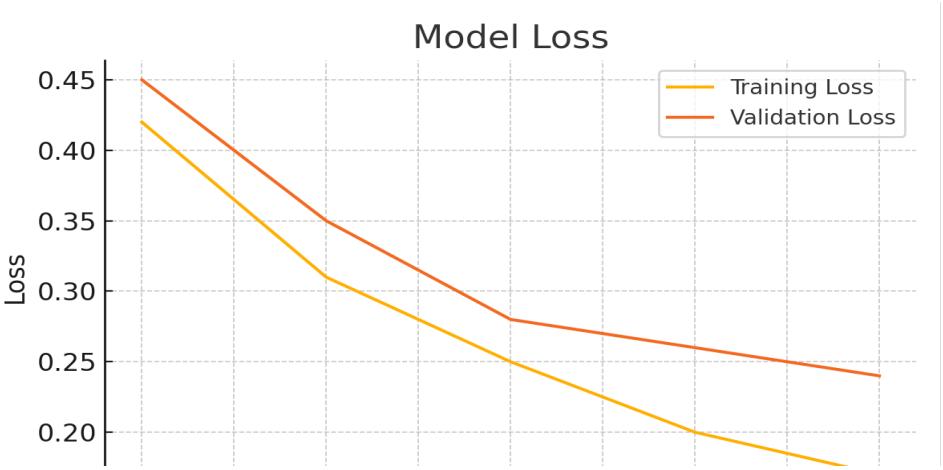

Fig. 6. Dataset.csv File Data Set

Figure 6 displays each line of SQL queries or injections that bypass the authentication or data and database extraction.

```
data = pd.read_csv("ml/dataset.csv")

# Очистка данных
def preprocess_query(query):  2 usages
    query = query.lower()
    query = re.sub(r"[\W_]+", " ", query)
    return query.strip()
```

Fig. 7. Bata Cleanup

SQL queries are preprocessed for stripping of all unwanted characters, normalization, and reducing to a single form (def preprocess_query (query):) as was demonstrated in Figure 6 previously and tokenization is done on the model to implement it on texts and turning the queries into sequence of numbers as in Figure 8 below.

```
# Токенизация
queries = data["query"].tolist()
tokenizer = Tokenizer()
tokenizer.fit_on_texts(queries)
sequences = tokenizer.texts_to_sequences(queries)
word_index = tokenizer.word_index
```

Fig. 8. Tokenization

After tokenization, each word in the query receives a unique index in Figure 8, but since the LSTM model is expecting input data of the same length, short queries are padded with zeros (padding=”post”), and long queries are truncated to max_length = 50:

```
# Заполняем последовательности
max_length = 50
padded_sequences = pad_sequences(sequences, maxlen=max_length, padding="post")
```

Fig. 9. The Input Array

After data preparation, the model is trained in LSTM architecture.

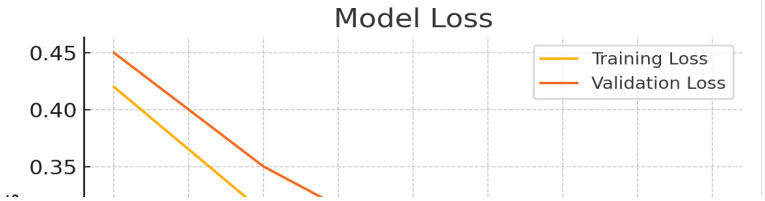

Fig. 10. Creating an LSTM Model

After finalizing the model architecture, it goes to the compilation and training stage, the neural network is designed in a way that it should be able to analyze a sequence of SQL queries and assign them labels as safe or potentially harmful.

The components of the model will have embedding layers, and these layers map the textual input into numerical vector representations as neural networks work with numbers and not raw texts. The model has two LSTM layers, and these are recurrent layers which sequentially process the tokenized input and help to identify temporal dependencies, the deeper they are with stacking the more complex the pattern they can learn. Finally, the output layer will use sigmoid activation which will provide a probability score between 0 and 1, describing the likelihood that the given SQL query is a security risk.

Model compiled with binary cross-entropy as loss function, as this is a binary class identification problem and Adam as optimizer.

The accuracy metric is used for the measurement of training performance as in Figure 11 below.

```
model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
```

Fig. 11. Model Components

The loss function used is “binary_crossentropy” which is the appropriate loss function for binary class problems as in the current case of identifying safe queries from unsafe queries.

The optimizer used is “adam” which is an adaptive learning rate optimization algorithm, this is to optimize the weights of the model parameters at faster rates during training

The monitoring metric used to track the model performance in classifying the inputs into correct classes during the training process is “accuracy”.

Having all these model configurations in place, the model is built, and the next stage is training.

```
# Обучение модели
model.fit(X_train, y_train, epochs=5, batch_size=32, validation_data=(X_test, y_test))
```

Fig. 12. Model Training

As shown in Figure 12 below, the setup for the fit method is as follows: X_train, y_train – training dataset and the labels against which the model will be trained Epochs = 5 – number of full iterations made by the model over the data during training batch_size = 32 – size of training samples made in every iteration before updating the model weights validation_data = (X_test, y_test) – this is the second dataset that will be used to test how the model is performing during training and also to watch for the overfitting problem.

```
# Оценка точности модели
y_pred = (model.predict(X_test) > 0.5).astype("int32")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Fig.13. Evaluating the Accuracy of the Model

After the training process is completed, the model's ability to classify SQL queries is evaluated. Having the model fully trained and tested, it will then be saved for later reuse without the need for retraining.

```
# Сохранение модели
model.save("ml/sql_injection_model.keras")
joblib.dump(tokenizer, "ml/tokenizer.pkl")
```

Fig. 14. Saving the Model

The image below shows the python code implementation for saving the already trained machine learning model of the used tokenizer. This step is a prerequisite for later reuse of the model without the need for retraining.

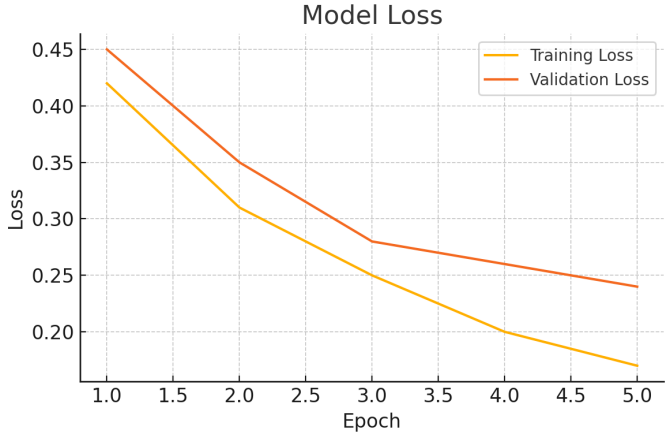

Fig. 15. Accuracy and Epoch Ratio ML Model

Figure 15 displays the Model accuracy graph showing how the accuracy metric for both training and validation set changed over the 5 training epochs.

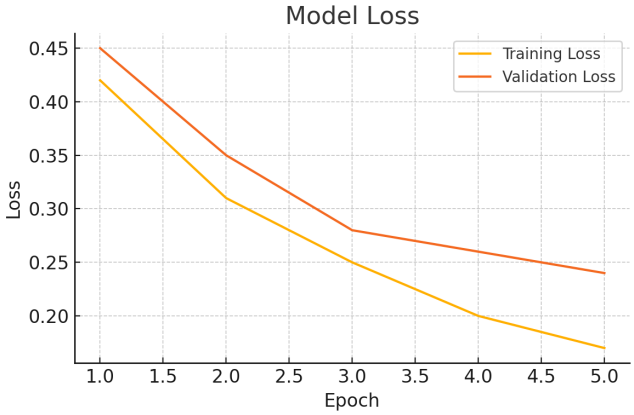

Fig. 16. Loss and Epoch Ratio ML Model

Figure 16 displays the Model loss graph which is showing how the loss metric for both training and validation set changed over the 5 training epochs.

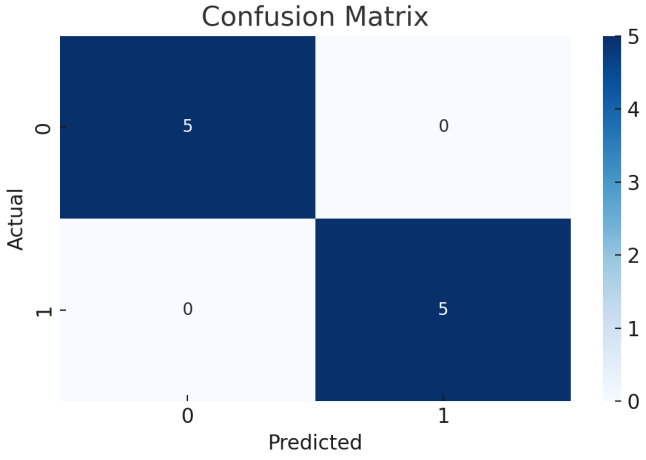

Fig. 17. Confusion Matrix

Figure 17 displays the confusion matrix table which is presenting the actual and predicted values on the training and testing results.

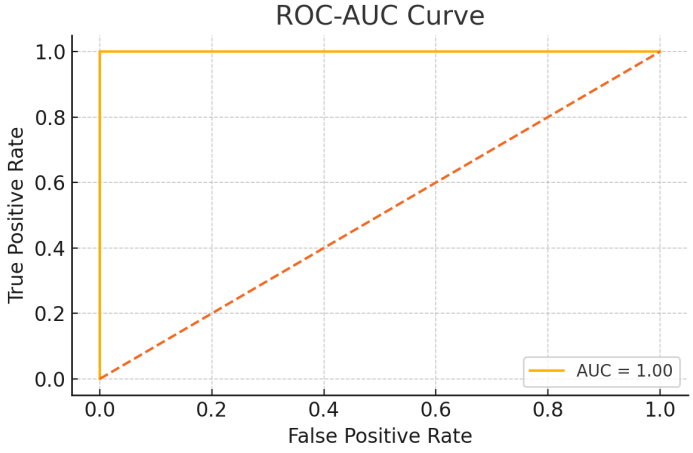

Fig.18. AUC-ROC Curve

Figure 18 displays the confusion matrix table presenting the true positive rate and false positive rate values on the training and testing results.

Conclusion

The system developed for the analysis of SQL queries using machine learning makes it possible to automatically detect potential threats such as SQL injections and analyze suspicious and abnormal user activity. During the project, a multi-level security system was implemented, which included authentication and access control, user activity monitoring, automatic SQL query analysis, and an alert system for suspicious activity.

The use of a combined SVM model allowed us to create an algorithm for classifying SQL queries and determining their security level. Optimization of the model and its integration into the server part on Python provided high accuracy for detecting potential threats.

Zabbix and Grafana were used to integrate for stable operation and control over the system, which allows you to track in real-time the number of and type of SQL queries being made as well as to identify potential attacks.

Project results:

- Automatic analysis of SQL queries with recommendations for fixing errors and improving security.
- Flexible authentication and access control system with JWT support.
- Intuitive React interface with the ability to view request history and data visualization.
- Real-time monitoring and logging with Zabbix and Grafana.
- High accuracy of SQL injection detection due to the use of machine learning.

The project is an effective solution for organizations that want to enhance the security of their databases and protect against potential attacks. Thanks to modern technologies, the system is easily scalable and adaptable for different use cases.

Acknowledgement. This research was carried out within the framework of the project BR24993014 "The development of an intelligent anti-corruption system for information protection, validation of the results of educational achievements, and official documents of students and graduates of universities in the Republic of Kazakhstan", which is being implemented at the Institute of Information and Computational Technologies.

REFERENCES

Any, Berlin, Tarisya Ramadhan, and Efa Ayu Nabila (2024). "Decentralized academic platforms: The future of education in the age of blockchain. Blockchain Frontier Technology 3. — No. 2. — 2024. —112–124.

Anwar, Farhat, Burhan Ul Islam Khan, M.L.B.M. Kiah, Nor Aniza Abdullah, and Khang Wen Goh (2022). "A comprehensive insight into blockchain technology: Past development, present impact and future considerations. — *International Journal of Advanced Computer Science and Applications* 13. — No. 11. — 2022.

Alsulami, Bader Muteb (2024). "Blockchain integrated data processing model for enabling security in Education 4.0. — Fusion: Practice & Applications 14. — No. 1. — 2024.

Boulet, Pierre, Perrine de Coëtlogon, Émilie Thébault, Patrice van de Velde, and Cyril Murie (2025). "Digital transformation of certificates issued by universities for European competitiveness." PhD diss., Université de Lille. — 2025.

Chen, Zhiyang, Hongwei Jiang, Jiapeng You, Xin Wang, and Poly ZH Sun (2024). "RFID lightweight authentication mechanism for smart factories based on blockchain. — IEEE *Journal of Radio Frequency Identification* 8. — 2024. — 19–30.

Capece, Guendalina, Nathan Levialdi Ghiron, and Francesco Pasquale (2020). "Blockchain technology: Re-

defining trust for digital certificates. — Sustainability 12. — No. 21. — 2020. — 8952.

Davletbayeva Zh.Zh., U.A. Bekish, A.V. Zagrebin and L.F. Raisova (2025). "ENHANCING THE ROLE OF PUBLIC COUNCILS AS AN ANTI-CORRUPTION MECHANISM: THE CASE OF KAZAKHSTAN. — 2025.

Eaton, Sarah Elaine, and Jamie J. Carmichael (2023). "Fake degrees and credential fraud, contract cheating, and paper mills: Overview and historical perspectives." Fake Degrees and Fraudulent Credentials in Higher Education. — 2023. — 1–22.

Gräther Wolfgang, Sabine Kolvenbach, Rudolf Ruland, Julian Schütte, Christof Torres, and Florian Wendland (2018). "Blockchain for education: lifelong learning passport." In Proceedings of 1st ERCIM Blockchain workshop 2018. European Society for Socially Embedded Technologies (EUSSET). — 2018.

Kustandi, Cecep, Muhamad Bakhar, Lila Pangestu Hadinigrum, Nina Oktarina, and Markus Asta Patma Nugraha (2024). "Utilizing blockchain technology for the advancement of education." In AIP Conference Proceedings. — Vol. 3098. — No. 1. — AIP Publishing. — 2024.

Li, Xiaoqi, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen (2020). "A survey on the security of blockchain systems." Future generation computer systems 107. — 2020. — 841–853.

Lu, Mingxuan, Zhichao Han, Susie Xi Rao, Zitao Zhang, Yang Zhao, Yinan Shan, Ramesh Raghunathan, Ce Zhang, and Jiawei Jiang (2022). "Bright-graph neural networks in real-time fraud detection." In Proceedings of the 31st ACM international conference on information & knowledge management. — Pp. 3342–3351. — 2022.

Li, Zoey Ziyi, K. Liu Joseph, Jiangshan Yu, and Dragan Gasevic (2022). "Blockchain-based solutions for education credentialing system: Comparison and implications for future development." In 2022 IEEE International Conference on Blockchain (Blockchain). — Pp. 79–86. — IEEE, 2022.

Panzabekova A., D. Fazylzhan and Z. Imangali (2024). "Analyzing the relationship between corruption and socio-economic development in Kazakhstan's regions." R-Economy. — Vol. 10. — Iss. 4 10. — No. 4. — 2024. — 455–474.

Rustemi, Avni, Fisnik Dalipi, Vladimir Atanasovski, and Aleksandar Risteski (2023). "A systematic literature review on blockchain-based systems for academic certificate verification." — IEEE Access 11. — 2023. — 64679–64696.

Tong Fei, Xing Chen, Kaiming Wang, and Yujian Zhang (2022). "CCAP: A complete cross-domain authentication based on blockchain for Internet of Things." IEEE Transactions on Information Forensics and Security 17. — 2022. — 3789–3800.

Turkanović, Muhamed, Marko Hölbl, Kristjan Košič, Marjan Heričko, and Aida Kamišalić (2018). "EduCTX: A blockchain-based higher education credit platform." — IEEE access 6. —2018. — 5112–5127.

Sharwani, Asif Ehsan, and Ramiro Melo (2024). "Blockchain-Enabled Academic Credential Verification in the USA." Adhyayan: — *A Journal of Management Sciences* 14. — No. 02. — 2024. — 31–41.

S. Mukhanov, O. Mamyrbayev, D. Katayev, A. Kalmurzayev, Zh. Oteuli, Sh. Yakupov, D. Amrin (2025). Analysis of the limits of model improvement in deep learning and performance saturation assessment, AIT 2025: — 1st International Workshop on Application of Immersive Technology. — March 5. — 2025. — Almaty. Kazakhstan.

S. Mukhanov, N. Komarov, O. Mamyrbayev, D. Amrin, Zh. Bolatov, I. Bazarbekov (2025). Analysis of modern problems in automatic speech recognition: solutions and practical examples, AIT 2025: 1st International Workshop on Application of Immersive Technology. — March 5. — 2025. — Almaty. Kazakhstan.

ХАЛЫҚАРАЛЫҚ АҚПАРАТТЫҚ ЖӘНЕ КОММУНИКАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАР ЖУРНАЛЫ

МЕЖДУНАРОДНЫЙ ЖУРНАЛ ИНФОРМАЦИОННЫХ И КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES

Правила оформления статьи для публикации в журнале на сайте:
https://journal.iitu.edu.kz

ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Собственник: АО «Международный университет информационных технологий» (Казахстан, Алматы)

Подписано в печать 15.09.2025.
Формат 60x881/8. Бумага офсетная. Печать - ризограф. 9,0 п.л. Тираж 100
050040 г. Алматы, ул. Манаса 34/1, каб. 709, тел: +7 (727) 244-51-09).

Издание Международного университета информационных технологий
Издательский центр КБТУ, Алматы, ул. Толе би, 59