Literature Review on SQL Injection in regards of AI

Christoph Hagenauer, Shaun Poole, Daniel Scheer

SQL Injection (SQLi) is a common cybersecurity vulnerability / attack vector. Our work provides an overview of recent developments in the field of SQLi prevention, detection, and response. Specifically, how SQLi prevention, detection, and response have evolved as Artificial Intelligence (AI) and Machine Learning Algorithms (ML) have been leveraged to combat SQLi attacks. A comprehensive overview of these topics as derived from analysis of 24 papers published in the past 4 years is provided (i.e., papers published since the advent of AL/ML), as is relevant historical background. Additionally, useful resources for novices in the field of cybersecurity are identified and shared.

I. Introduction

Structured Query Language (SQL) is an international standard developed to interact and store data on a server that can be accessed through a network connection to view and manipulate the stored data [10]. The SQL standard is a way to describe the relation between data points. The data can then be retrieved through a network based on the previously defined relations in the form of a query to the Database Management System (DBMS). The DBMS then retrieves and returns the data from the database according to the relation provided in the query [9, 12].

SQL injection (SQLi) is when a threat actor inputs malicious query patterns into an application with the goal of accessing restricted data. By inserting certain patterns into the SQL query, the threat actor can extract and infer information about the DBMS, structure of the database, and the underlying data. [3, 17, 4] According to the Open Worldwide Application Security Project's (OWASP) most recent ”OWASP Top Ten” survey, SQLi ranked in third place only being outranked by authentication bypasses and cryptographic failures. ”The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications.” [13]

Historically, detecting SQLi has relied on suspicious patterns in SQL queries or network statistics. Building on that and the fact that machine learning and AI has developed at an immense rate. We have seen AI/ML algorithms being developed and applied to classify injection attempts. Various methods have been researched and tested, and some are being used in industry today. [4]

Taking all this into consideration this paper aims to provide a balanced introduction to the subject of SQLi and the AI and ML methods that are being applied to combat injections today. Furthermore, we provide a brief discussion about the evolution of SQLi detection. We will attempt to achieve this by providing recent and relevant references to literature relating to all the subjects we will be discussing as well as highlight what we believe to be the most important parts of the field today that need to be further developed through research.

This paper starts by explaining different SQLi attacks in III. In IV we give a brief overview of the evolution of SQLi detection prevention. In V we go in more detail about current approaches. This is followed by a discussion VI about different models and current limitations we discovered.

II. Methodology

A. Identification of Research Objective

With this research we want to give our peers a starting point for SQLi detection methods. Therefore, we came up with the following research questions:

- RQ1: How has SQL injection attack detection evolved throughout the years?
- RQ2: What SQLi attacks exist?
- RQ3: What are some novel SQLi detection frameworks?

B. Searching the Literature

To conduct our research, we searched Databases from Elsevier Science Direct, MDPI, Springer Link, Research Gate, and Google Scholar. For this search we used the following search words:

SQL Injection; AI; Structured Query Language Injection; Artificial Intelligence; ML; Machine Learning; DL; Deep Learning; SQL Detection; Structured Query Language Detection

C. Paper Inclusion Criteria

Selected papers meet the following criteria:

- Paper has to be relevant to SQL Injection
- Paper has to clearly outline techniques used for SQL Injection detection / prevention.
- The paper includes an empirical evaluation of the proposed method using real-world or simulated datasets.
- The paper must outline what AI/ML algorithm/algorithms are applied to achieve the resulting performance metrics.
- Paper has to be clearly structured and easy to follow.
- Paper has to be published after 2022; except cross-references from papers dealing with evolutions of SQLi detection, as well as definitions in general.
- Paper has to be peer-reviewed

D. Paper Exclusion Criteria

We are not going to consider any publications which do not meet the following criteria:

- Papers published before 2022, due to the fact of technology being very fast moving.

- Papers not published electronically.
- Papers that are not peer reviewed.

E. Snowballing

After reviewing relevant literature reviews published after 2022, we used the snowballing research technique to examine each paper's reference list extracting what we believe to be important. Out of those, we collected all relevant papers that met our inclusion/exclusion criteria and ended up with **X**.

F. Data Extraction & Analysis

To evaluate the most prominent research directions we perform a word analysis on the papers using python. We used ChatGPT to generate a python script to parse through the papers, conducting word counts, and create resulting word clouds and word count graphs. The python script is attached in

III. Attack Types

All SQLi attacks have the same core idea, inserting malicious patterns into a input field of some type in order to trick the DBMS to send data back to the requester that they are not authorized to view. [3, 4, 17]

For the purpose of this paper we give an overview of eight SQLi attack types based on how they attempt to extract information from the database.

- **Tautology Query** : The threat-actor injects conditions that would always evaluate to true, causing the entire WHERE clause to always return true resulting in the DBMS sending back information even if the programmed condition is false.
- **Illegal Query** : The threat-actor sends an illegal / logically incorrect query to the DBMS. This can typically only happen if there is not enough input sanitation in the application.
- **Union Query** : The threat actor attempts to use the UNION operator that joins two query results with the same number of columns together into one returned table.
- **Piggy Backed Query** : The threat-actor sends two queries to the database, the first one is a ”legitimate” query to the database followed by a malicious query.
- **Blind Query** : The threat-actor continually queries the database asking it true or false conditions to infer information about the DBMS based on the responses.
- **Timing Query** : The threat-actor tries to infer information from the database through a specified timing programmed in to the query. If the malicious query is true the DBMS will respond after the programmed timing.
- **Stored Procedure Query** : The threat-actor attempts to inject malicious statements into the database through the parameters of a stored procedure.
- **Alternate Encoding Query** : The threat-actor uses different numerical representations of special characters to evade special character filters.

The definitions of these attacks are based on the following two papers [14]

IV. Evolution Of SQLi Detection

SQLi detection started with static pattern matching, which would look for suspicious queries. The goal was to identify malicious SQL code by performing regular expression searches (REGEX) and checking the input against a predefined set of patterns. Furthermore, There have traditionally been two approaches to input validation, blacklist and whitelist approaches were used to find and filter illegal queries from the DBMS [6]. The next evolution of SQLi detection was Machine Learning Algorithms [19, 18, 7, 8]. Machine Learning is the practice of algorithmically regressing or categorizing predictions based on a set of provided features in the data; Typically a model is trained on historic data to identify patterns within the dataset. Machine Learning is a subset of Artificial Intelligence which enables computers to learn form historic datasets. This information is used to make predictions on future data Fairly soon after Deep learning (DL) was introduced to SQLi detection [16, 1]. DL is an extension of machine learning to extract more complex patterns from larger datasets in order to bring machine learning closer to the ultimate goal of Artificial intelligence. Deep Learning Networks are able to find complex structures in large datasets by simulating the interconnecting structure of human brains. The fundamental difference between machine learning and deep learning is that the performance of deep learning as the size of the data increases. [16, 15].

V. Current approaches applied to SQLi detection

In this section we highlight two novel papers from late 2024 and early 2025. We focus on the datasets, encoding techniques, and proposed frameworks used by these papers.

A. GenSQLi

In [2], the authors created a framework which used Large Language Models (LLMs) to both generate and defend against SQLi attacks [2]. By using LLMs to generate SQLi attacks, the attacks were quickly written, obfuscated, and varied. They used this to pinpoint holes in a Web Application Firewall (WAF) which can safeguard a DB or application.

To generate the SQLi attacks, the authors prompted ChatGPT-4o and Gemini-Pro with 11 manually crafted SQLi attacks. In response, the models generated a total of 907 SQLi attacks which varied in attack type and complexity. The attacks were then validated by sending them through a vulnerable PHP application which used a simple MySQL database. This step weeded out any queries having various errors which could not be executed on the DB.

Once validated, the SQLi attacks were sent through a a WAF to simulate a real attack against an application. The attacks that made it through the WAF were then identified and classified using two machine learning algorithms: TF-IDF with Hierarchal Agglomerative Clustering (HAC) alongside SequenceMatcher with DBSCAN. TF-IDF emphasizes rare attack-specific terms while reducing the weights of common terms while HAC effectively handles irregular, hierarchal relationships. SequenceMatcher captures string-based similarities, and DBSCAN manages overlapping clusters, noise tolerance, and dynamic identification of cluster count.[2] The classified attacks were then used to prompt GPT-4o to generate new firewall rules to prevent any future attacks. The initially generated rules contained various issues which were corrected through Reinforcement Learning with Human Feedback. As a result, the model produced rules with fewer errors, broader coverage of SQLi patterns, and reduced false positives. [2]

It's worth mentioning this framework can be configured to work with various applications or DB setups; However, it must be applied from the very start of the framework. Otherwise, the initial LLMs will produce language specific syntax that isn't supported by other versions of SQL. For example, MySQL contains some defined functions that are not found in PostgreSQL or Microsoft SQL. This limitation combined with the various points of human intervention can be prevent this framework from widespread use in the industry.

B. Light-Bert

The other framework **BERT-improvement** uses a dataset with more than 100,000 entries, which are slightly skewed in favor of SQLi attacks; approximately 60% of queries are malicious 40% are regular queries. The BERT-paper needs this amount of data because they are training their model, rather than interfacing with an existing model hosted by a third party like the GenSQLi's approach.

In [11] the authors remove all query data except SQL keywords or SQL special characters by employing a SQL-specific tokenizer when preprocessing inputs to the model. Specifically, they use their own SQL vocabulary to filter the query that is used as input to the model. The authors implemented this so only parts relevant to detecting the maliciousness of the query is considered when trying to classify the query. Once the query data is filtered it is split by keyword and named a skeleton key representing the structure of the query. That skeleton key is then tokenized to produce two embedded matrices, one representing semantic meaning and another representing token embedding. The matrices are then augmented and multiplied by a position matrix that creates the input matrix to the model. This has the advantage of boosting malicious characteristics of the input while ensuring that the model can use self-attention to be able to extract context from the rest of the query.

The goal of Lo, Hwang, and Tai [11] was to build a more computationally efficient model than the existing approaches being applied. By doing this, they enable the model to reside on less powerful hardware, which means the model can be run locally on the client and would be close to the application. Therefore, the I/O overhead that is needed to transmit the query to a more powerful machine is eliminated, and latency is reduced. In the framework, after the encoding step, the SQLi payload is evaluated by a Convolutional Neural Network (CNN), which they are using to extract features from the input matrix. Afterwards, the SQLi payload (convoluted from previous layer) is evaluated by a transformer and finally classified using a Dense layer feeding into a single node using a sigmoid activation function. [11].

In their paper [11], the authors improve upon the BERT model [5], which enables the model to reside on less powerful hardware. The main emphasis of the paper is on the importance of feature extraction out of SQLi payloads and to retain high accuracy and performance metrics while requiring less computational overhead to enable client-side payload analysis. This ultimately allows the framework to be more power efficient and more flexible in its application.

VI. Discussion

In summary, a significant amount of papers we looked at were using the same dataset performing similar analysis. However, we did notice several emerging algorithms and / or frameworks such as those highlighted above. Grounded in ML / DL methods, these algorithms and / or frameworks are able to extend the capabilities of the traditional methods to detect such patterns. We observed an emerging threat coming from Chat tools (such as ChatGPT) being able to write potentially undetectable SQLi patterns; this highlights the need for new defense mechanisms [2].

We want to emphasize the importance of encoding. Encoding is essential in preparing input data to ML / DL models, as the model performance heavily depends on the encoding. Another very important step is input validation, which was dismissed by many papers in favor of testing their proposed models. Omitting input validation and other security measures significantly decreases the security of systems and their omission during testing leads to misrepresentation of the model's applicability in real-world systems.

While we believe this technology adds significant security measures the main conclusion is to maintain focus on writing secure code! Many SQLi attacks can be prevented by implementing traditional detection and prevention methods, as well as adhering to security practices.

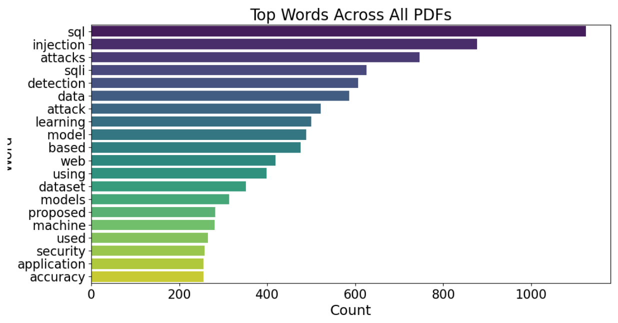

Fig. 1. Word Cloud

References

[1] Maha Alghawazi, Daniyal Alghazzawi, and Suaad Alarifi. "Deep learning architecture for detecting SQL injection attacks based on RNN autoencoder model". In: *Mathematics* 11.15 (2023), p. 3286.

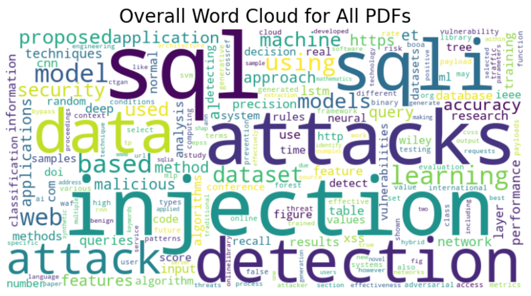

Fig. 2. Word Cloud

[2] Vahid Babaey and Arun Ravindran. "GenSQLi: A Generative Artificial Intelligence Framework for Automatically Securing Web Application Firewalls Against Structured Query Language Injection Attacks." In: *Future Internet* 17.1 (2025).

[3] Ding Chen et al. "SQL Injection Attack Detection and Prevention Techniques Using Deep Learning". In: *Journal of Physics: Conference Series*. Vol. 1757. 2021. DOI: 10.1088/1742-6596/1757/1/012055.

[4] Justin Clarke-Salt. *SQL injection attacks and defense*. Elsevier, 2009.

[5] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186.

[6] Hsiu Chuan Huang et al. "Web Application Security: Threats, Countermeasures, and Pitfalls". In: *Computer* 50 (6 2017). ISSN: 00189162. DOI: 10.1109/MC.2017.183.

[7] Anamika Joshi and V. Geetha. "SQL Injection detection using machine learning". In: *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies, ICCICCT 2014*. 2014. DOI: 10.1109/ICCICCT.2014.6993127.

[8] Krit Kamtuo and Chitsutha Soomlek. "Machine learning for SQL injection prevention on server-side scripting". In: *20th International Computer Science and Engineering Conference: Smart Ubiquitos Computing and Knowledge, ICSEC 2016*. 2017. DOI: 10.1109/ICSEC.2016.7859950.

[9] Jagpreet Kaur and K. R. Ramkumar. "The recent trends in cyber security: A review". In: *Journal of King Saud University - Computer and Information Sciences* 34 (8 Sept. 2022), pp. 5766–5781. ISSN: 22131248. DOI: 10.1016/j.jksuci.2021.01.018.

[10] Brad Kelechava. *The SQL Standard - ISO/IEC 9075:2023 (ANSI X3.135) - ANSI Blog*. https://blog.ansi.org/sql-standard-iso-iec-9075-2023-ansi-x3-135/. [Accessed 16-04-2025].

[11] Rui-Teng Lo, Wen-Jyi Hwang, and Tsung-Ming Tai. "SQL Injection Detection Based on Lightweight Multi-Head Self-Attention". In: *Applied Sciences* 15.2 (2025), p. 571.

[12] Abhishek Munnolimath et al. *SQL Standards*. en-US. reference. URL: https://docs.oracle.com/en/database/oracle/oracle-database/23/sqlrf/SQL-Standards.html#GUID-92B38403-6934-4E86-9D9A-E94957ACDDFC (visited on 04/16/2025).

[13] OWASP Foundation. *OWASP Top Ten*. Accessed: 2025-04-16. 2021. URL: https://owasp.org/www-project-top-ten/.

[14] Edwin Peralta-Garcia et al. "Detecting structured query language injections in web microservices using machine learning". In: *Informatics*. Vol. 11. 2. MDPI. 2024, p. 15.

[15] Luis Serrano. *Grokking machine learning*. Simon and Schuster, 2021.

[16] Peng Tang et al. "Detection of SQL injection based on artificial neural network". In: *Knowledge-Based Systems* 190 (2020), p. 105528.

[17] HR Yasith Wimukthi et al. "A comprehensive review of methods for SQL injection attack detection and prevention". In: *International Journal of Scientific Research in Science and Technology IJSRST* (2022).

[18] Xi Rong Wu and Patrick P.K. Chan. "SQL injection attacks detection in adversarial environments by K-centers". In: *Proceedings - International Conference on Machine Learning and Cybernetics*. Vol. 1. 2012. DOI: 10.1109/ICMLC.2012.6358948.

[19] Wei Dong Zhao, Wei Hui Dai, and Chun Bin Tang. "K-centers algorithm for clustering mixed type data". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 4426 LNAI. 2007. DOI: 10.1007/978-3-540-71701-0_129.