Journal of Cybersecurity and Privacy

MDPI

Article

Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review

Maha Alghawazi, Daniyal Alghazzawi and Suaad Alarifi *

Information Systems Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 80200, Saudi Arabia

* Correspondence: salarifi@kau.edu.sa

Abstract: An SQL injection attack, usually occur when the attacker(s) modify, delete, read, and copy data from database servers and are among the most damaging of web application attacks. A successful SQL injection attack can affect all aspects of security, including confidentiality, integrity, and data availability. SQL (structured query language) is used to represent queries to database management systems. Detection and deterrence of SQL injection attacks, for which techniques from different areas can be applied to improve the detect ability of the attack, is not a new area of research but it is still relevant. Artificial intelligence and machine learning techniques have been tested and used to control SQL injection attacks, showing promising results. The main contribution of this paper is to cover relevant work related to different machine learning and deep learning models used to detect SQL injection attacks. With this systematic review, we aims to keep researchers up-to-date and contribute to the understanding of the intersection between SQL injection attacks and the artificial intelligence field.

Keywords: SQL injection; machine learning; deep learning; adversarial attacks

Citation: Alghawazi, M.; Alghazzawi, D.; Alarifi, S. Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review. *J. Cybersecur. Priv.* **2022**, *2*, 764–777. https://doi.org/10.3390/jcp2040039

Academic Editor: Marina L. Gavrilova

Received: 31 July 2022
Accepted: 14 September 2022
Published: 20 September 2022

1. Introduction

Most cyber-physical system (CPS) applications are safety-critical; misbehavior caused by random failures or cyber-attacks can considerably restrict their growth. Thus, it is important to protect CPS from being damaged in this way [1]. Current security solutions have been well-integrated into many networked systems including the use of middle boxes, such as antivirus protection, firewall, and intrusion detection systems (IDS). A firewall controls network traffic based on the source or destination address. It alters network traffic according to the firewall rules. Firewalls are also limited to their knowledge of the hosts receiving the content and the amount of state available. An IDS is a type of security tool that scans the system for suspicious activity, monitors the network traffic, and alerts the system or network administrator [2]. In this context, a number of frameworks and mechanisms have been suggested in recent papers.

In this paper, we have considered SQL injection attacks that target the HTTP/HTTPS protocol, which aim to pass through the web application firewall (WAF) and obtain an unauthorized access to proprietary data. SQL injection belongs to the injection family of web attacks, wherein an attacker inserts inputs into a system to execute malicious statements. The victim system is usually not ready to process this input, typically resulting in data leakage and/or granting of unauthorized access to the attacker; in this case, the attacker can access and/or modify the data, affecting all aspects of security, including confidentiality, integrity, and data availability [3].

In an SQL injection, the attacker inserts an SQL statement into an exchange between a client and database server [3]. SQL (structured query language) is used to represent queries to database management systems (DBMSs). The maliciously injected SQL statement is designed to extract or modify data from the database server. A successful injection can result

in authentication and bypass and changes to the database by inserting, modifying, and/or deleting data, causing data loss and/or destruction of the entire database. Furthermore, such an attack could overrun and execute commands on the hosted operating system, typically leading to more serious consequences [4].

Thus, SQL injection attacks present aserious threats to organizations. A variety of research has been undertaken to address this threat, presenting various artificial intelligence (AI)techniques for detection of SQL injection attacks using machine learning and deep learning models [5]. AI techniques to facilitate the detection of threats are usually implemented via learning from historical data representing an attack and/or normal data. Historical data are useful for learning, in order to recognize patterns of attacks, understanding detected traffic, and even predicting future attacks before they occur [6].

SQL injection attackers and defenders must understand how SQL language works to know how it can be misused [3]. To extract data from a database or modify the data, queries must be written using SQL language and they must follow a standard syntax, such as:

"SELECT * FROM books WHERE author = 'MAHA'"

The above query will return all books authored by MAHA. Queries are submitted to the DBMS and are usually written through a web browser. For the query to be transmitted to the database server through the web browser, it has to be encoded through a long URL string, such as: http://www.xyz_website.com?QUERY=SELECT%20*%20FROM%20books%20WHERE%20author=7453.

What if the attacker adds to the previous SQL query? For example:

"SELECT * FROM books WHERE author = ′MAHA′ OR ′1′ = ′1′"

As the statement 1 = 1 is always true, the query will return all books in the database, not just the books authored by MAHA.

The previous example might not represent a threat, especially if the stored list of books is not confidential. However, it could be applied to valuable using different syntax, and if successful, it might return sensitive data, such as passwords, bank accounts, trade secrets, and personal data, which might be considered a privacy breach, among other consequences.

In some research, injecting a code using 'OR' followed by a TRUE statement, such as 1 = 1 is called "tautology" [7]. Methods other than tautology can be used, such as when an attacker intentionally injects an incorrect query to force the database server to return a default error page, which might contain valuable information that could help an attacker to understand the database to form a more advance attack [7]. The SQL syntax "UNION" can also be used to extract information, in addition to many other methods based on the same idea, of misusing SQL syntax to extract or even update the data in the targeted database.

This is how SQL injection works; the question then becomes: how does one detect this type of attack using deep learning methods?

Deep learning is a machine learning and artificial intelligence method. It can be used to support the detection of SQL injection attacks by training a classifier to achieve the ability to recognize and therefore detect an attack. The classifier is trained using deep learning models and can be used to classify new data, such as traffic or data in log files. If the classifier is passive, it will alert the administrator; if it is active, it will prevent data from passing to the database server. The classifier can be trained to recognize and detect SQL injection attacks using three different learning methods [8].

First is, unsupervised learning, where features are extracted from unclassified data, i.e., data that are labelled as neither normal nor abnormal. Using information and the Bayesian probability theory, the classifier detects hidden structures in the unclassified dataset. An unclassified dataset means that it is not known whether these data are normal or abnormal (malicious). Different techniques can be used in unsupervised learning, such as clustering and density estimation [8].

The second is, supervised learning, whereby a labelled training dataset is used to train the classifier. As the input data are labelled, i.e., normal or abnormal, the output is known beforehand. Therefore, the process involves simple mapping between the input training data and the known output, followed by continuous modification of the algorithm and changing of the weights until an acceptable classification accuracy is achieved. Then, a test dataset is used to test the classifier; if the result is with an acceptable accuracy range, the classifier is ready to detect novel data, i.e., data not previously used in training or testing. The main drawback of supervised learning is generating and labelling the training and testing data, which might consume processing time, especially for complex attacks. Supervised learning is categorized into classification and regression algorithms. The most common supervised learning algorithms include Bayesian networks, decision trees, support vector machines (SVMs), K-nearest neighbors, and neural networks. Third is, semi-supervised learning, which use combination of supervised and unsupervised learning methods [8].

The main contribution of this paper is to provide a systematic review of the machine learning and deep learning solutions that, are used to improve the detectability of SQL injection attacks. With this systematic review, we aim to keep researchers up-to-date and contribute to the understanding of the intersection between an SQL injection attack and artificial intelligence.

The paper is organized as follows. Section 1 is an introduction to SQL injection attacks and deep learning algorithms. In Section 2, we discuss related studies and consider previous systematic reviews. In Section 3, we present the research method and planning of the systematic review. In Section 4, highlights the results and review all related studies. In Section 5, presents the discussion and answers to research questions. Finally, in Section 6, we present our conclusions.

2. Related Studies

In this section, four published systematic reviews were considered. Newer systematic reviews typically include both recent and older studies in the area under investigation. Therefore, all of the papers we considered were relatively recent. The first was published in 2017 [9] and it covered previous primary studies on SQL injection attacks, techniques, and tools. In [9], forty-six primary studies were analyzed related to SQL injection attacks, tools, and techniques, in addition to the impact of the attack. We adapted the same methodology as that used in [9] due to its comprehensiveness and because it achieves satisfying results, in addition, this research was similar to that in [9] in terms of objectives, ideas, and the area of research.

Qiu et al. [10] provided a comprehensive review of using artificial intelligence in attacking and defending against security attacks, concentrating on the training and testing stages. In their study, they sorted technologies and applications of adversarial attacks in terms of natural language processing, cyberspace security, computer vision, and the physical world. Furthermore, the authors considered defense strategies in their research and proposed methods to deal with specific types of adversarial attack. Martins et al. [11] explored more than 15 papers that applied adversarial machine learning techniques used in intrusion and malware detection models. In their study, the authors summarized the most common adversarial attacks and defense mechanisms for intrusion and malware detection.

Muslihi et al. [12] conducted a review of more than 14 studies published using deep learning methods to detect SQL injection attacks, including CNN, LSTM, DBN, MLP, and Bi-LSTM. They also provided a comparison of methods in terms of objectives, techniques, features, and datasets. Muhammad et al. [13] reviewed and analytically evaluated the methods and tools that are commonly used to detect and prevent SQL injection attacks, considering a total of 82 studies. Their review results showed that most researchers focused on proposing approaches to detect and mitigate SQL injection attacks (SQLIAs) rather than evaluating the effectiveness of existing SQLIA detection methods.

3. Research Method

This systematic literature review was conducted in four main phases: (A) planning the systematic review; (B) conducting the review; (C) reporting the results; and (D) discussing the results. In the planning phase, research questions and the research strategy were set. Section 4 outlines the systematic review. We discuss our results in Section 5. Figure 1 is a representation of the phases of this research.

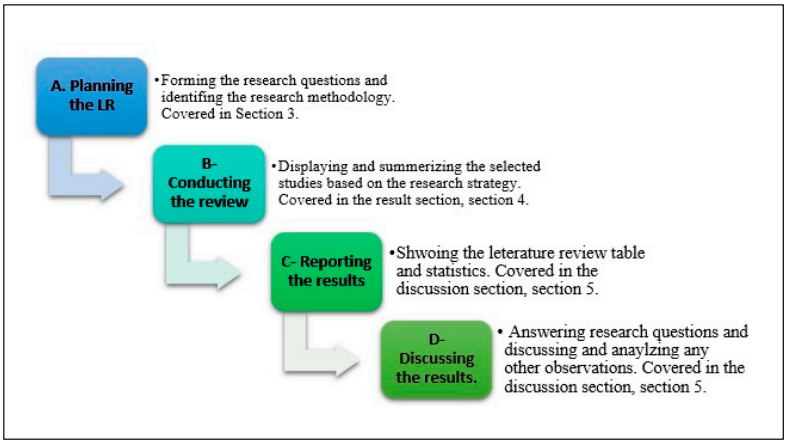

Figure 1. Research phases.

3.1. Planning the Systematic Review

Research Questions

Q1: What are the machine learning and deep learning methods used to detect SQL injection attacks?

Q2: How are SQL injection attack datasets generated using machine learning techniques?

Q3: How can machine learning be used to generate adversarial SQL injection attacks?

The first question was the main question decided upon before starting the review, whereas the second and third questions were added later after reviewing other systematic reviews covered in Section 4.

3.2. Research Strategy

The libraries used to retrieve the research papers were ACM, IEEE, Springer and Science Direct. The main search topics were SQL injection attacks and machine learning models. The search was configured to retrieve papers published between 2012 and 2021, and we retrieved conference papers, journal articles, and review articles. Some inclusion criteria were defined to select relevant papers among the publications retrieved at the time of the search. These criteria were used to decide which papers to review and which to discard and not include for further study.

3.2.1. Inclusion Criteria

- Papers related to SQL injection attacks;
- Papers that included our search keywords;
- Papers from the scientific databases ACM, IEEE, SpringerLink, and ScienceDirect.
- Papers on the topic of machine learning and the security domain.

3.2.2. Exclusion Criteria

- Papers not covering machine learning techniques and SQL injection attacks;
- Papers published before 2012; and
- Papers that are not available in full-text format.

4. Results

Conducting the Review

After filtering retrieved studies according to the inclusion criteria, 36 studied were retained. Selected studies were reviewed, as they could possibly provided answers to the research questions.

Q1: What are the machine learning and deep learning methods used to detect SQL injection attacks?

Many researchers have demonstrated the use of machine learning and deep learning algorithms to detect SQL injection attacks [14]. Hasan and Tarique [14] tested and compared 23 machine learning classifiers using MATLAB. They generated their own datasets, into which they injected abnormal SQL syntax. They checked and manually verified the SQL statements. A total of 616 SQL statements were used to train the test classifiers. The used the following machine learning algorithms: "coarse k-NN, bagged trees, linear SVM, fine k-NN, medium k-NN, RUS boosted trees, subspace discriminant, boosted trees, weighted k-NN, cubic k-NN, linear discriminant, medium tree, subspace k-NN, simple tree, quadratic discriminant, cubic SVM, fine Gaussian SVM, cosine k-NN, complex tree, logistic regression, coarse Gaussian SVM, medium Gaussian, and SVM". The five best models in terms of accuracy were determined to be ensemble boosted, bagged trees, linear discriminant, cubic SVM, and fine Gaussian SVM.

Gao et al. [15] proposed a model called ATTAR to detect SQL injection attacks by analyzing web access logs to extract SQL injection attack features. The features were chosen based on access behavior mining and a grammar pattern recognizer. The main target of this model was detection of unknown SQL injection statements that had not been previously used in the training data. Five machine learning algorithms were used for training: naive Bayesian, random forest, SVM, ID3, and k-means. The experimental results showed that the accuracy of the models based on random forest and ID3 achieved the best results in detecting SQL injection attacks. We could not find what ATTAR stands for in [15].

Gandhi et al. [16] proposed a hybrid CNN-BiLSTM-based model for SQL injection attack detection. The authors presented a detailed comparative analysis of different types of machine learning algorithms used for detection of SQL injection attacks. The CNN-BiLSTM approach provided accuracy of approximately 98%, compared withother described machine learning algorithms.

Zhang [17] presented a machine learning classifier to detect SQL injection vulnerabilities in PHP code. Multiple machine learning algorithms were trained and evaluated, including random forest, logistic regression, SVM, multilayer perceptron (MLP), long short-term memory (LSTM), and a convolutional neural network (CNN). Zhang found that CNN provided the best precision of 95.4%.

Gi Li et al. [18] proposed an adaptive deep forest model (ADF) with the integration of the AdaBoost algorithm. AdaBoost stands for adaptive boosting, which is a statistical classification algorithm, and the deep forest model is a layered model based on a deep neural network. The adaptive deep forest model proposed in [16] achieved high efficiency, comparable to that of traditional machine learning models, such as decision trees, and a better performance compared with regular deep neural network models, such as RNN and CNN.

Uwagbole et al. [19] created a dataset using symbolic finite automata to train a classifier to detect SQL injection attacks. The generated data were labelled, and training was conducted with a supervised learning model with an ML algorithm of two-class support vector machine (TC SVM) and two-class logistic regression (TC LR). The generated models were evaluated using a receiver operating characteristic (ROC) curve.

Ahmed et al. [20] proposed an SQL injection detection method using an ensemble learning algorithm and natural language processing (NLP) to generate a bag-of-words model used to train a random forest classifier. Prediction was also considered in this research to improve the detection ability of the classifier. In this study, decision tree, naïve Bayes, SVM, and k-NN classification models were also trained to classify the same testing dataset, and their performances were compared with that of the proposed method. The

experimental results showed that the proposed method achieved better accuracy, higher TPR, and lower FNR than the other four classifiers. Evaluation metrics were used to measure the performance of the classifier. The measurements were based on a confusion matrix, accuracy, precision, true-positive rate, false-positive rate, true-negative rate, false-negative rate, receiver operating characteristic curve, and area under the curve.

Tripathy et al. [21] created a dataset by gathering and combining a large number of smaller datasets. The generated dataset was labelled, and the learning model was supervised learning. They trained seven machine learning models: decision tree, AdaBoost, random forest, optimized linear, TensorFlow linear, deep ANN, and a boosted trees classifier. Then, they compared the seven algorithms in terms of performance and accuracy. The results showed that the random forest classifier outperformed all other classifiers and achieved an accuracy of 99.8%.

Chinmay and Kulkarni [22] proposed a novel approach to detection of SQL injection attacks using a human agent knowledge transfer (HAT) and TD machine learning algorithm. In this model, a machine learning agent acted as a maze game to differentiated between normal SQL queries and malicious SQL queries. If the incoming SQL query was an SQL injection attack query, then it gained more rewards and was deemed an SQL injection attack query before achieving the final state. This machine learning approach achieved an accuracy of 95%.

Makiou et al. [23] proposed a detection system based on two approaches. The first detection method was based on pattern matching, which is the same as a signature-based detection system whereby the classifier has a database of SQL attack signatures and only inspects the HTTP URL in an attempt to find a match. The second detection method used was based on machine learning techniques. To build this model, the authors collected malicious data and trained the classifier with these data by extracting the features representing attacks. The following algorithms were employed: SVM, naïve Bayes, and K-nearest neighbor. The performance of the classifier was measured using the total cost ratio (TCR).

Kar et al. [24] trained a support vector machine (SVM) to detect malicious SQL queries by modelling the WHERE clause of a query as an interaction network of tokens and computing the centrality of the nodes. Node centralities were used to quantify the degree of importance or centrality of a node in the network. The experimental results obtained on a dataset collected from five web applications using some automated attack tools, confirmed that three of the centrality measures used in this study can effectively detect SQL injection attacks with minimal impact on performance.

Wang et al. [25] analyzed the existing SQL injection detection algorithms in an intelligent transportation system. The authors proposed a long short-term memory (LSTM)-based SQL injection attack detection method and a method of generating SQL injection samples to augment the dataset. This method can simulate SQL injection attacks and generate valid positive samples to solve the problem of overfitting caused by a lack of positive samples. The experimental results showed that the accuracy, precision, and F1 score of the proposed method were all above 92%.

Kamtuo and Soomlek. [26] proposed a framework for SQL injection prevention via server-side scripting using machine learning and compiler platforms. A dataset of 1100 samples of SQL commands were trained in four machine learning models: boosted decision tree, decision tree, support vector machine (SVM), and an artificial neural network. The results indicate that the decision tree algorithm achieved the highest prediction efficiency among the tested models.

Sivasangari et al. [27] used the AdaBoost algorithm to detect SQL injection attacks. In this study, the data were converted into stumps, which were classified as weak stumps providing less weight to the output or strong stumps providing the highest weight in the overall output. The experimental result showed that the proposed algorithm accurately and effectively detected injection attacks.

Daset al. [28] proposed a method for classifying dynamic SQL queries as either attacks or normal based on a web profile prepared during the training phase. Naïve Bayes, SVM,

and parse tree approaches were used for the classification process. The overall detection rate using the two datasets was 91% and 90%, respectively.

Kasim [29] designed a method to detect malicious SQL queries. Decision tree algorithms were used for the classification processes to detect different levels of SQL injection. The proposed model maintained an accuracy more than 98% in detecting SQL injection attacks and an accuracy of 92% in classifying the level of attack as simple, unified, or lateral.

Tanget et al. [30] presented a simple method for SQL injection attack detection based on an artificial neural network. First, a large amount of SQL injection data were analyzed to extract the relevant features. Then, a variety of neural network models, such as MLP and LSTM, were trained. The experimental results showed that the detection rate of MLP was better than that of LSTM.

Erdődiet al. [31] automatized the process of exploiting SQL injection attacks through reinforcement learning agents. In this study, the problem was modelled as a Markov decision process. The experimental results show that reinforcement learning agents can be used in the future to perform security assessment and penetration testing.

Kar et al. [32] presented a detection method by modeling SQL queries as a graph of tokens and utilized the centrality measure of tokens to train single and multiple SVM classifiers. The system was tested using directed and undirected graphs with different SVM classifiers. The experimental results demonstrated that the proposed technique is able to effectively identify malicious SQL queries.

Solomon et al. [33] presented a model of a two-class support vector machine (TCSVM) to predict binary labelled outcomes concerning whether an SQL injection attack was positive or negative in a web request. This model intercepted web requests at the proxy level and applied ML predictive analytics to predict SQL injection attacks.

Mcwhirter et al. [34] presented a novel approach for classifying SQL queries. A gap-weighted string subsequence kernel algorithm was used to compute the similarity metric between the query strings. Then, the support vector machine was trained on the similarity metrics to determine whether the query strings was normal or malicious. The proposed approach was evaluated using a number of datasets and achieved 92.48% accuracy.

Mejia-Cabrera et al. [35] presented a new approach to the construction of a dataset with a NoSQL query database. Six classification algorithms were trained and evaluated to identify SQL injection attacks, which included: decision tree, SVM, random forest, k-NN, neural network, and multilayer perceptron. The experimental results showed that the last two algorithms obtained an accuracy of 97.6%.

Pathak et al. [36] trained a progressive neural network model with a naïve Bayesian classifier to successfully detect SQL injection attacks. Progressive neural networks were trained using parameters such as error-based, time-based, SQL query and, union-based SQL injection attacks. The proposed method achieved an accuracy of 97.897%.

Wang et al. [37] proposed a hybrid approach using tree-vector kernels in SVM to learn SQL statements. The authors used both the parse tree structure of SQL queries and the query value similarity characteristic to distinguish between malicious and benign queries. The results confirmed the benefit of incorporation to efficiently and accurately identify abnormal queries.

Fang et al. [38] proposed a tool based on LSTM neural networks and the word vectors of SQL tokens. According to the syntactic functions of the SQL queries, each query was converted into sequences of tokens to build an SQL word vector model. Then, the LSTM neural network was trained. The results of the experiment showed that the proposed tool achieved an accuracy of 98.60%.

Zhang et al. [39] proposed a deep learning-based approach to detect SQL injection attacks in network traffic. The proposed approach selected only the target features needed by the model to be trained using a deep belief network (DBN) model. The authors also employed test data to test the performance of different models, including LSTM, CNN, and MLP. According to the experimental results, DBN achieved an accuracy of 96%.

Priyaa et al. [40] proposed a framework that combined the EDADT (efficient data adaptive decision tree) algorithm and the SVM classification algorithm to detect SQL injection attacks. The employed dataset was created using the MovieLens dataset system for movie recommendations, which included user login and movie details. The experimental results showed that the proposed approach achieved an accuracy of 99.87%.

Joshi et al. [41] proposed a method for detecting SQL injection using the naïve Bayes machine learning algorithm. The authors applied a tokenization process to break the query into meaningful elements called tokens. Then, the list of tokens became an input for the further classification processes. The result of the naïve Bayes approach was analyzed using precision, recall, and accuracy.

Q2: How are SQL injection attack datasets generated using machine learning techniques?

Many researchers have been developed and generated their SQL injection datasets instead of using existing datasets [42]. Islam et al. [43] developed a training dataset for NoSQL injection to manually design important features using various supervised learning algorithms. In this study, the authors generated a dataset including approximately 75% benign and 25% injection queries, which was tested on a local server.

Appelt et al. [44] proposed automated testing techniques that generated SQL injection attacks, bypassing web application firewalls (WAFs). The authors developed SQL injection grammar based on existing SQL injection attacks, as well as an automated input generation technique to automatically generate attack payloads. Then, machine learning was used to efficiently generate additional payloads and new successful attacks with a high probability of bypassing the firewall.

Ross et al. [42] proposed a system consisting of three phases to generate data: traffic generation, capture, and preprocessing. In the traffic generation phase, the simulated normal and malicious traffic was generated from the scripts located on the traffic generation server. Then, the traffic was captured by the webapp server and at the Datiphy appliance. Finally, data preprocessing was achieved with bash shell scripts on the webapp server. The resulting data from preprocessing was imported into Weka, which is a machine learning framework that includes many ML tools. The data were processed into word vectors using the weak filter StringToVec. Then correlated feature selection was employed to reduce the number of features for efficient machine learning.

Liu et al. [45] proposed a tool called DeepSQLi to generate test cases for detection of SQL injection attacks using a deep learning model and sequence-of-words prediction. DeepSQLi used the neural language model, which can be trained to learn semantic features of SQL attacks to translate the test case (or user input) into a new test case. Therefore, DeepSQLi is able to generate SQL injection attacks that have not been captured by patterns in the training datasets. Siddiq et al. [46] proposed a learning-based SQL injection fix tool called SQLIFIX. This tool creates an abstraction of SQL injection code from a training dataset that consists of 14 projects and then clusters them using hierarchical clustering. The proposed approach generated correct solutions for 67.52% of cases for Java and 41.33% of correct solutions for PHP on an independent test set.

Naghmeh [47] proposed a model for the detection of SQLI attacks using artificial intelligence (AI) techniques. This model consisted of three main components: uniform resource locator (URL) generator to generate thousands of normal and malicious URLs; a URL classifier to classify all generated URLs as either normal or malicious; and a neural network (NN) model to detect whether a given URL was a malicious, or benign URL. The model was first trained and then evaluated by employing both benign and malicious URLs. URL classifiers were also used to convert all generated URLs into strings of logic (1 = malicious; 0 = benign).

Q3:How can machine learning be used to generate adversarial SQL injection attacks?

Adversarial machine learning (AML) is based on the threats posed by an attacker with the aim of being incorrectly classified by the victim machine learning algorithm. Generating an adversarial SQL injection dataset starts with a target malicious query that was correctly

detected. And then, a set of mutation operators was iteratively applied in order to generate new queries [48].

Demetrio and Valenza [48] developed a tool named WAF-A-MoLE to generate adversarial examples against web application firewalls (WAFs) by applying a set of syntactic mutations. The authors produced a dataset of SQL injection queries through an automatic procedure. To evaluate the effectiveness of the proposed tool, it was applied to different ML-based WAFs and evaluated in terms of their robustness against WAF-A-MoLE.

Appelt et al. [49] proposed a black-box automated technique, named 4SQLi, for generating test inputs that could bypass security filters, resulting in executable SQL queries. This technique was based on a set of multiple mutation operators that manipulated inputs to produce new test inputs to trigger SQLi attacks, making it possible to create inputs that contained new attack patterns, thus increasing the possibility of generating a successful SQLi attacks.

5. Discussion

5.1. Machine Learning and Deep Learning Techniques for Detection of SQL Injection Attacks (Related to Q1)

In this section, the results reported in Section 4 are discussed. In related studies, various algorithms and techniques can be used for detecting SQL injection attacks. Table 1 summarizes the algorithms under review, in addition to the employed datasets and evaluation methods.

Table 1. Summary of the Machine learning algorithms, Datasets, and Evaluation Methods.

Ref.	Algorithm	Dataset	Dataset Size	Evaluation Methods										
				Accuracy	FPR		FNR		TP	FN	FP	TN	Precision	Recall	F1 Score	AUC
[13]	Naïve Bayesian	Collected from access logs	58,000 log records	-	10.9%	16.7%	34.5%	18.2%	-	-	-	-	-	-	-	-
	SVM			-	4.1%	8.3%	41.4%	18.2%	-	-	-	-	-	-	-	-
	ID3			-	0.0%	0.0%	41.4%	18.2%	-	-	-	-	-	-	-	-
	RF			-	0.68%	0.0%	37.9%	9.1%	-	-	-	-	-	-	-	-
	K-means			-	0.68%	0.0%	37.9%	9.1%	-	-	-	-	-	-	-	-
[14]	CNN-BiLSTM	Collected from various websites	4200 queries (3072 SQL injections,1128 normal data	98%	-		-		-	-	-	-	-	-	-	-
[15]	Decision Tree	Collected from two sources	950 vulnerable PHP cases, 8800 non-vulnerable files	93.4%	-		-		-	-	-	-	76.6%	56.5%	0.650%	-
	Random Forest			93.6%	-		-		-	-	-	-	77.4%	57.7%	0.660%	-
	SVM			95.4%	-		-		-	-	-	-	98.6%	58.3%	0.732%	-
	Logistic Regression			95.1%	-		-		-	-	-	-	98.5%	56.0%	0.713%	-
	Multilayer Perceptron			95.3%	-		-		-	-	-	-	91.0%	63.7%	0.746%	-
	RNN			95.3%	-		-		-	-	-	-	92.2%	62.4%	0.742%	-
	LSTM			95.2%	-		-		-	-	-	-	91.9%	61.4%	0.734%	-
	CNN			95.3%	-		-		-	-	-	-	95.4%	59.9%	0.734%	-
[16]	ADF	Collected from vulnerability submission platforms	10,000 negative samples and 10,000 positive samples	Not clear	-		-		-	-	-	-	-	-	-	-
	AdaBoost															
[17]	Two-Class Logistic Regression		Dataset of 725,206 attribute values	96.4%	-		-		-	-	-	-	0.971	0.957	0.964	0.984
	Two-Class Support Vector Machine			98.6%	-		-		-	-	-	-	0.974	0.998	0.986	0.986
[18]	Random Forest + NLP	Open-source tools, such as Libinjection and Sqlmap	17,266 thousand SQL injection payloads and 19,303 thousand normal payloads	98.1515	0.96137		0.03862		4182	168	1	4792	0.9997%	-	-	0.99
[19]	RF	Collected from datasets available in public repositories	7576 malicious SQL queries and 100,496 legal inputs	99.8%	-		-		-	-	-	-	0.999	0.999	0.999	-
	TensorFlow Boosted Trees Classifier			99.6%	-		-		-	-	-	-	0.989	0.961	0.998	-
	AdaBoost Classifier			99.5%	-		-		-	-	-	-	0.997	0.996	0.997	-
	Decision Tree			99.5%	-		-		-	-	-	-	0.998	0.997	0.997	-
	SGD Classifier			98.6%	-		-		-	-	-	-	0.988	0.997	0.992	-
	Deep ANN			98.4%	-		-		-	-	-	-	0.934	0.820	0.873	-
	TensorFlow Linear Classifier			97.8%	-		-		-	-	-	-	0.908	0.759	0.988	-

Table 1. *Cont.*

Ref.	Algorithm	Dataset	Dataset Size	Evaluation Methods										
				Accuracy	FPR	FNR	TP	FN	FP	TN	Precision	Recall	F1 Score	AUC
[12]	Ensemble Boosted Trees	Open-source datasets	616 SQL statements	93.8%	-	-	-	-	-	-	-	-	-	-
	Bagged Trees			93.8%	-	-	-	-	-	-	-	-	-	-
	Linear Discriminant			93.7%	-	-	-	-	-	-	-	-	-	-
	Cubic SVM			93.7%	-	-	-	-	-	-	-	-	-	-
	Gaussian SVM			93.5%	-	-	-	-	-	-	-	-	-	-
[20]	TD Machine Learning Technique	Not mentioned	Not mentioned	95%.	-	-	-	-	-	-	-	-	-	-
[21]	SVM, Naïve Bayes, K-Nearest Neighbor	Open-source datasets	Not mentioned	Not clear	-	-	-	-	-	-	-	-	-	-
[22]	SVM classifier	Dataset generated using a honeypot-based technique.	4610 injected and 4884 genuine token sequences	92.84%	1.33%	86.66%	914	8	8	969	98.40%	86.66%	-	-
				99.16%	0.82%	99.13%	799	123	13	964	99.13%	99.31%	-	-
				99.37%	0.72%	99.46%	917	5	7	970	99.24%	99.46%	-	-
				99.05%	1.02%	99.13%	914	8	10	967	98.92%	99.13%	-	-
[23]	LSTM	Open-source datasets	Not mentioned	93.47%	-	-	-	-	-	-	93.56%	92.43%	92.99%	
[24]	SVM, Boosted Decision Tree, Artificial Neural Network, Decision Tree	Open-source datasets	1100 vulnerable SQL commands	99.68%	-	-	-	-	-	1.000	-	-	-	-
[25]	AdaBoost algorithm	Not mentioned	Not mentioned	Not Clear	-	-	-	-	-	-	-	-	-	-
[26]	Naïve Bayesian	Not mentioned	Not mentioned	90%	-	-	-	-	-	-	-	-	-	-
	SVM			91%										
	Parse Tree			91%										
[27]	Decision tree	OWASP dataset	332 malicious codes and 52 the clean codes	98%	-	-	-	-	-	97%	98%	97%		98.2%
[28]	MLP	Open-source datasets	820 SQL injection samples and 8925 normal samples	99.67%	0.00%	-	-	-	-	100%	99.41%	-	-	-
	LSTM			97.68%	0.13%	-	-	-	-	99.86%	95.49%	-	-	-
[29]	Markov Decision Processes (MDPs)	Not mentioned	1000 SQL environments	-	-	-	-	-	-	-	-	-	-	-
[30]	SVM	Open-source datasets	4610 injected queries and 4884 genuine queries	99.37%	0.31%	-	-	-	-	-	99.35%	99.35%	99.46%	-
				99.73%	0.31%	-	-	-	-	-	99.67%	99.78%	99.73%	-
				99.63%	0.31%	-	-	-	-	-	99.67%	99.57%	99.62%	-
[31]	TCSVM	Dataset from MicrosoftSQL reserved keywords website	362,603 attack items and 362,603 non-attack items	98.60%	-	-	-	-	-	-	97.4%	99.7%	98.5%	98.6%
[32]	SVM	Amnesia testbed dataset	46 legitimate queries and 40 malicious SQL injection attacks	-	-	-	-	-	-	-	65.9%	98.3%	78.9%	-
											68%	100%	81%	
[33]	Support Vector Machine	Novel datasets	450 malicious and 59 benign queries	84.9%	-	-	-	-	-	-	84.8%	91.1%	87.6%	83.3%
	K-Nearest Neighbor			87.6%	-	-	-	-	-	-	84.8%	96.7%	90.4%	96.6%
	Neural Network			97.6%	-	-	-	-	-	-	98.7%	97.4%	98.0%	98.9%
	Multilayer Perceptron			97.6%	-	-	-	-	-	-	98.7%	97.4%	98.0%	98.9%
	Decision Tree			89.4%	-	-	-	-	-	-	96.3%	85.6%	90.6%	94.6%
	Random Forest			89.6%	-	-	-	-	-	-	87.5%	96.4%	91.7%	97.4%
[34]	Progressive Neural Network, Naïve Bayes	Open-source dataset	A 62.2 KB SQL query and a 4.86 KB SQL injection exploitation	97.897%	-	-	193	0	0	5	-	-	-	-
[35]	SVM	Open-source dataset	1000 benign and 1000 malicious HTTP requests		0.982	0.000	-	-	-	-	-	-	-	-
[36]	LSTM	Open-source dataset	43,167 injected query strings and 32,486 genuine query strings	98.60%	-	-	-	-	-	-	99.17%	99.20%,	99.17%	99%
[37]	LSTM, MLP, CNN, Deep Belief Network (DBN)	Datasets collected from HTTP requests	118,529 normal data points and 21,810 SQL injection data points		-	-	-	-	-	-	-	-	-	-
[38]	EDADT and SVM	Dataset created based on the MovieLens dataset	Not mentioned	99.87%	-	-	-	-	-	-	-	-	-	-
[39]	Naïve Bayes	Not mentioned	101 normal codes and 77 malicious codes	93.3%	-	-	-	-	-	-	1.0	0.89	-	-

Table 1 shows that most of the studies focused on using supervised machine learning to detect and classify SQL injection attacks; 89% of the studies used supervised learning, and 4% used unsupervised learning and mixed learning, whereas 3% used other types of learning, as shown in Figure 2.

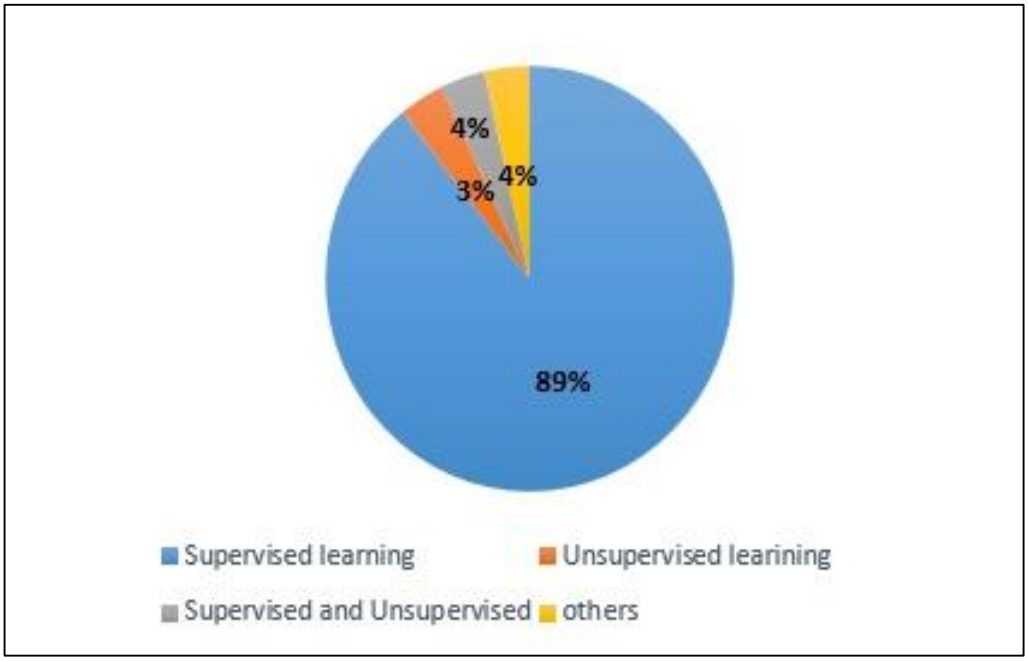

Figure 2. Percentage of the mchine learning and deep learning techniques used idetecting SQL injection attacks.

5.2. Generating SQL Injection Attack Datasets Using Machine Learning Techniques (Related to Q2)

A high-quality dataset for training is essential for machine learning and deep learning methods to achieve effective detection performance. It is difficult to identify suitable datasets with patterns to train classifiers in SQL injection attack research [30]. The results of the studies reviewed in Section 4 showed that, after automatically generating SQL injection attack payloads from different web applications, machine learning techniques can learn incrementally learn the payloads that are passed or blocked by the firewalls and can be used to efficiently generate additional payloads with high probability of bypassing the firewall. A total of 83% of the reviewed studies used datasets collected from public repositories and HTTP requests. The remaining 17% of the reviewed studies used datasets created by the authors using deep learning models that can be trained to learn the semantic features of SQL attacks to generate new test cases from user inputs.

5.3. Generating Adversarial SQL Injection Attacks Using ML Techniques (Related to Q3)

The result reported in Section 4 showed that adversarial SQL injection attacks can be generated using mutation operators, which are a set of operators that alter the syntax of the original payload without affecting its semantics. Such operators can be classified into three classes based on their purpose: behavior-changing, syntax-repairing, and obfuscating operators [49,50]. Table 2 provides a summary of the mutation operators.

Table 2. Summary of mutation operators (adopted from [50]).

MO Class	MO Name	Description	Example
Behavior-Changing Operators	MO or	Adds an OR clause to the input	Original input: "SELECT * FROM table WHERE id= " the input will change the logic of the statement and turns it as follows: "SELECT * FROM table WHERE id = 1 OR 1 = 1
	MO and	Adds an AND clause to the input	
	MO semi	Adds a semicolon followed by an additional clause	
Syntax-Repairing Operators	MO par	Appends a parenthesis to a valid input	Original inpt: "SELECT * FROM table WHERE character = CHR(" + input + ")" The changed SQL statement: SELECT * FROM table WHERE character = CHR(67) OR 1 = 1 {).
	MO cmt	Adds a comment command (– or #) to an input	
	MO qot	Adds a single or double quote to an input	
Obfuscating Operators	MO wsp	Changes the encoding of white spaces	Original input: 1 OR 1 = 1, mutated input: 1+− OR + 1 = 1. This changes the predefined statement: "SE- LECT * FROM table WHERE id = " + input to SELECT * FROM table WHERE id = 1 + OR + 1 = 1
	MO chr	Changes the encoding of a character literally enclosed in quotes	
	MO html	Changes the encoding of an input to HTML entity encoding	
	MO per	Changes the encoding of an input to percentage encoding	
	MO bool	Rewrites a Boolean expression while preserving its truth value	
	MO keyw	Obfuscates SQL keywords by randomizing the capitalization and inserting comments	

6. Conclusions

SQL injection attacks represent a major threat to web applications, and this may have major implications for privacy and security. Machine learning and deep learning applications have achieved considerable success in detecting this type of web attack. In this study, we conducted a systematic literature review of 36 articles related to research on SQL injection attacks and machine learning techniques. We identified the most commonly used machine learning techniques to detect all types of SQL injection attacks. The review results showed that few studies used machine learning tools and methods to generate new SQL injection attack datasets. Similarly, the results showed that only a few studies focused only on using mutation operators to generate adversarial SQL injection attack queries. In future work, we aim to cover the use of other machine learning and deep learning models to generate and detect SQL injection attacks., In addition to investigating the use of other AI techniques to generate adversarial SQL injection attacks, such as generative adversarial networks (GANs).

Author Contributions: Conceptualization, M.A. and D.A.; methodology, M.A.; software, M.A.; validation, M.A., D.A. and S.A.; formal analysis, O.R.; investigation, M.A.; resources, M.A.; data curation, M.A.; writing—original draft preparation, M.A.; writing—review and editing, S.A.; visualization, M.A.; supervision, D.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under Grant No. IFPDP-284-22. The authors, therefore, acknowledge with thanks to DSR technical and financial support.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Han, S.; Xie, M.; Chen, H.-H.; Ling, Y. Intrusion Detection in Cyber-Physical Systems: Techniques and Challenges. *IEEE Syst. J.* **2014**, *8*, 1049–1059. [CrossRef]
2. Mishra, P.; Varadharajan, V.; Tupakula, U.; Pilli, E.S. A Detailed Investigation and Analysis of using Machine Learning Techniques for Intrusion Detection. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 686–728. [CrossRef]
3. Charles, M.J.; Pfleeger, P.; Pfleeger, S.L. *Security in Computing*, 5th ed.; Springer: Berlin/Heidelberg, Germany, 2004.

4. Son, S.; McKinley, K.S.; Shmatikov, V. Diglossia: Detecting code injection attacks with precision and efficiency. *Proc. ACM Conf. Comput. Commun. Secur.* **2013**, *2*, 1181–1191. [CrossRef]
5. Yan, R.; Xiao, X.; Hu, G.; Peng, S.; Jiang, Y. New deep learning method to detect code injection attacks on hybrid applications. *J. Syst. Softw.* **2018**, *137*, 67–77. [CrossRef]
6. Vähäkainu, P.; Lehto, M. Artificial intelligence in the cyber security environment. In Proceedings of the 14th International Conference on Cyber Warfare and Security, ICCWS 2019, Stellenbosch, South Africa, 28 February–1 March 2019; pp. 431–440.
7. Satapathy, S.C.; Govardhan, A.; Raju, K.S.; Mandal, J.K. SQL Injection Detection and Correction Using Machine Learning Techniques. *Adv. Intell. Syst. Comput.* **2015**, *337*, 435–442. [CrossRef]
8. Marashdeh, Z.; Suwais, K.; Alia, M. A Survey on SQL Injection Attacks: Detection and Challenges. In Proceedings of the 2021 International Conference on Information Technology (ICIT), Amman, Jordan, 14–15 July 2021; pp. 957–962. [CrossRef]
9. Faker, S.A.; Muslim, M.A.; Dachlan, H.S. A systematic literature review on sql injection attacks techniques and common exploited vulnerabilities. *Int. J. Comput. Eng. Inf. Technol.* **2017**, *9*, 284–291.
10. Qiu, S.; Liu, Q.; Zhou, S.; Wu, C. Review of artificial intelligence adversarial attack and defense technologies. *Appl. Sci.* **2019**, *9*, 909. [CrossRef]
11. Martins, N.; Cruz, J.M.; Cruz, T.; Abreu, P.H. Adversarial Machine Learning Applied to Intrusion and Malware Scenarios: A Systematic Review. *IEEE Access* **2020**, *8*, 35403–35419. [CrossRef]
12. Muslihi, M.T.; Alghazzawi, D. Detecting SQL Injection on Web Application Using Deep Learning Techniques: A Systematic Literature Review. In Proceedings of the 2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE), Surabaya, Indonesia, 3–4 October 2020. [CrossRef]
13. Aliero, M.S.; Qureshi, K.N.; Pasha, M.F.; Ghani, I.; Yauri, R.A. Systematic Review Analysis with SQLIA Detection and Prevention Approaches. *Wirel. Pers. Commun.* **2020**, *112*, 2297–2333. [CrossRef]
14. Hasan, M.; Tarique, M. Detection of SQL Injection Attacks: A Machine Learning Approach. In Proceedings of the 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, 19–21 November 2019.
15. Gao, H.; Zhu, J.; Liu, L.; Xu, J.; Wu, Y.; Liu, A. Detecting SQL Injection Attacks Using Grammar Pattern Recognition and Access Behavior Mining. In Proceedings of the 2019 IEEE International Conference on Energy Internet (ICEI), Nanjing, China, 27–31 May 2019. [CrossRef]
16. Gandhi, N. A CNN-BiLSTM based Approach for Detection of SQL Injection Attacks. In Proceedings of the 2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, 17–18 March 2021; pp. 378–383.
17. Zhang, K.; Dataset, A.T. A Machine Learning based Approach to Identify SQL Injection Vulnerabilities. In Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA, 11–15 November 2019; pp. 2019–2021. [CrossRef]
18. Li, Q.I.; Li, W.; Wang, J. A SQL Injection Detection Method Based on Adaptive Deep Forest. *IEEE Access* **2019**, *7*, 145385–145394. [CrossRef]
19. Uwagbole, S.O.; Buchanan, W.J.; Fan, L. An Applied Pattern-Driven Corpus to Predictive Analytics in Mitigating SQL Injection Attack. In Proceedings of the 2017 Seventh International Conference on Emerging Security Technologies (EST), Canterbury, UK, 6–8 September 2017; pp. 12–17.
20. Ahmed, M. Cyber Attack Detection Method Based on NLP and Ensemble Learning Approach. In Proceedings of the 2020 23rd International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 19–21 December 2020; pp. 19–21.
21. Tripathy, D.; Gohil, R.; Halabi, T. Detecting SQL Injection Attacks in Cloud SaaS using Machine Learning. In Proceedings of the 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), Baltimore, MD, USA, 25–27 May 2020; pp. 145–150. [CrossRef]
22. Kulkarni, C.C.; Kulkarni, S.A. Human agent knowledge transfer applied to web security. In Proceedings of the 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Tiruchengode, India, 4–6 July 2013; pp. 14–17. [CrossRef]
23. Makiou, A.; Begriche, Y.; Serhrouchni, A. Hybrid approach to detect SQLi attacks and evasion techniques. In Proceedings of the 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, Miami, FL, USA, 22–25 October 2014; pp. 452–456. [CrossRef]
24. Kar, D.; Sahoo, A.K.; Agarwal, K.; Panigrahi, S.; Das, M. Learning to Detect SQLIA Using Node Centrality with Feature Selection. In Proceedings of the 2016 International Conference on Computing, Analytics and Security Trends (CAST), Pune, India, 19–21 December 2016; pp. 18–23.
25. Li, Q.; Wang, F.; Wang, J.; Li, W. LSTM-Based SQL Injection Detection Method for Intelligent Transportation System. *IEEE Trans. Veh. Technol.* **2019**, *68*, 4182–4191. [CrossRef]
26. Kamtuo, K.; Soomlek, C. Machine Learning for SQL Injection Prevention in Server-Side Scripting. In Proceedings of the 2016 International Computer Science and Engineering Conference (ICSEC), Chiang Mai, Thailand, 14–17 December 2016; pp. 1–6.
27. Sivasangari, A. SQL Injection Attack Detection using Machine Learning Algorithm. In Proceedings of the 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 3–5 June 2021; pp. 1166–1169.

28. Das, D.; Sharma, U.; Bhattacharyya, D.K. Defeating SQL injection attack in authentication security: An experimental study. *Int. J. Inf. Secur.* **2019**, *18*, 1–22. [CrossRef]
29. Kasim, Ö. An ensemble classification-based approach to detect the attack level of SQL injections. *J. Inf. Secur. Appl.* **2021**, *59*, 102852. [CrossRef]
30. Tang, P.; Qiu, W.; Huang, Z.; Lian, H.; Liu, G. Detection of SQL injection based on artificial neural network. *Knowl.-Based Syst.* **2020**, *190*, 105528. [CrossRef]
31. Erdődi, L.; Sommervoll, Å.Å.; Zennaro, F.M. SQL injection vulnerability exploitation using Q-learning reinforcement learning agents. *J. Inf. Secur. Appl. Simulating* **2021**, *61*, 102903. [CrossRef]
32. Kar, D.; Panigrahi, S.; Sundararajan, S. SQLiGoT: Detecting SQL injection attacks using the graph of tokens and SVM. *Comput. Secur.* **2016**, *60*, 206–225. [CrossRef]
33. Uwagbole, S.O.; Buchanan, W.J.; Fan, L. Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention. In Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, 8–12 May 2017; pp. 1087–1090. [CrossRef]
34. Mcwhirter, P.R.; Kifayat, K.; Shi, Q.; Askwith, B. SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel. *J. Inf. Secur. Appl.* **2018**, *40*, 199–216. [CrossRef]
35. Mejia-Cabrera, H.I.; Paico-Chileno, D.; Valdera-Contreras, J.H.; Tuesta-Monteza, V.A.; Forero, M.G. *Automatic Detection of Injection Attacks by Machine Learning in NoSQL Databases*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 23–32.
36. Pathak, R.K.; Yadav, V. *Handling SQL Injection Attack Using Progressive Neural Network*; Springer: Singapore, 2020; Volume 1170.
37. Wang, Y.; Li, Z. SQL injection detection via program tracing and machine learning. In *Lecture Notes in Computer Science*; 7646 LNCS; Springer: Berlin/Heidelberg, Germany, 2012; pp. 264–274. [CrossRef]
38. Fang, Y.; Peng, J.; Liu, L.; Huang, C. WOVSQLI: Detection of SQL injection behaviors using word vector and LSTM. In Proceedings of the ICCSP 2018: Proceedings of the 2nd International Conference on Cryptography, Security and Privacy, Guiyang, China, 16–19 March 2018; pp. 170–174. [CrossRef]
39. Zhang, H.; Zhao, J.; Zhao, B.; Yan, X.; Yuan, H.; Li, F. SQL injection detection based on deep belief network. In Proceedings of the CSAE 2019: Proceedings of the 3rd International Conference on Computer Science and Application Engineering, Sanya, China, 22–24 October 2019. [CrossRef]
40. Priyaa, B.D.; Student, P.G.; Devi, M.I. Hybrid SQL Injection Detection System. In Proceedings of the 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 22–23 January 2016.
41. Joshi, A. SQL Injection Detection using Machine Learning. In Proceedings of the 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kanyakumari, India, 10–11 July 2014; Volume 2, pp. 1111–1115.
42. Ross, K.; Moh, M.; Yao, J.; Moh, T.S. Multi-source data analysis and evaluation of machine learning techniques for SQL injection detection. In Proceedings of the ACMSE 2018 Conference, Richmond, KY, USA, 29–31 March 2018; pp. 1–8. [CrossRef]
43. Islam, M.R.U.; Islam, M.S.; Ahmed, Z.; Iqbal, A.; Shahriyar, R. Automatic detection of NoSQL injection using supervised learning. In Proceedings of the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 15–19 July 2019; Volume 1, pp. 760–769. [CrossRef]
44. Appelt, D.; Nguyen, C.D.; Briand, L. Behind an application firewall, are we safe from SQL injection attacks? In Proceedings of 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST), Graz, Austria, 13–17 April 2015. [CrossRef]
45. Liu, M.; Li, K.; Chen, T. DeepSQLi: Deep semantic learning for testing SQL injection. In Proceedings of the ISSTA 2020: Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, 18–22 July 2020; pp. 286–297. [CrossRef]
46. Siddiq, M.L.; Jahin, R.R.; Rafid, M.; Islam, U. SQLIFIX: Learning-Based Approach to Fix SQL Injection Vulnerabilities in Source Code. In Proceedings of the 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Honolulu, HI, USA, 9–12 March 2021; pp. 354–364. [CrossRef]
47. Sheykhkanloo, N.M. Employing Neural Networks for the detection of SQL injection attack. In Proceedings of the SIN '14: Proceedings of the 7th International Conference on Security of Information and Networks, Glasgow, UK, 9–11 September 2014; pp. 318–323. [CrossRef]
48. Demetrio, L.; Valenza, A.; Costa, G.; Lagorio, G. WAF-A-MoLE: Evading web application firewalls through adversarial machine learning. In Proceedings of the SAC '20: Proceedings of the 35th Annual ACM Symposium on Applied Computing, Brno, Czech Republic, 30 March–3 April 2020; pp. 1745–1752. [CrossRef]
49. Appelt, D.; Nguyen, C.D.; Briand, L.C.; Alshahwan, N. Automated testing for SQL injection vulnerabilities: An input mutation approach. In Proceedings of the 2014 International Symposium on Software Testing and Analysis, San Jose, CA, USA, 21–25 July 2014; pp. 259–269.
50. Appelt, D. Automated Security Testing of Web-Based Systems against SQL Injection Attacks. Ph.D. Thesis, University of Luxembourg, Luxembourg, 2016.