DEVELOPING SQL INJECTION PREVENTION MODEL USING DEEP LEARNING TECHNIQUE

A Thesis Presented

by

Abenezer Ketema

to

The Faculty of Informatics

of

St. Mary's University

In Partial Fulfillment of the Requirements

for the Degree of Master of Science

in

Computer Science

July, 2022

ACCEPTANCE

Developing SQL Injection Prevention Model Using Deep Learning Technique

By

Abenezer Ketema

Accepted by the Faculty of Informatics, St. Mary's University, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Thesis Examination Committee:

____

Internal Examiner

Dr.Alembante Mulu

____

External Examiner

Dr. Melkamu Hunegnaw(PhD) [illegible]

____

Dean, Faculty of Informatics

Dr.Alembante Mulu

June 23, 2022

DECLARATION

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other university, and all sources of materials used for the thesis work have been duly acknowledged.

Abenezer Ketema

____

Signature

Addis Ababa

Ethiopia

This thesis has been submitted for examination with my approval as an advisor.

____

Advisor: Dr. Minale Ashager (PhD)

____

Signature

Addis Ababa, Ethiopia

June 23, 2022

Acknowledgment

First and foremost, I would like to thank the almighty God for his divine guidance and constant support throughout this study. I would also like to express my deepest and most sincere gratitude to my advisor, Minale Ashagre (PhD.), for his tireless efforts in helping me with this study. Additionally, I would like to thank him for his continuous support of my MSc research, his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all aspects of my research and writing of this thesis. I could not have imagined finishing my thesis without his support and encouragement.

I am extremely grateful to Natnael Tilahun, and Ashenafi Worku Wondimu for giving me their comments and suggestions to complete the thesis. And I would like to thank all staff members of St. Mary University and the Department of Computer Science in particular for their enormous assistance. Last but certainly not least, I would like to express my deepest gratitude to my parents for helping, supporting, and encouraging me not only within this paperwork process but also throughout my life.

Table of Contents

List of Acronyms and Abbreviations

ANN	**:**Artificial Neural Network
CONF	**:**Confusion
DBMS	**:** Database Management System
DL	**:** Deep Learning
Metr	**:** Metrics
ML	**:** Machine Learning
RQ	**:** Research Questions
SQL	**:** Structural Query Language.
SQLIA	**:** Structural Query Language Injection Attack
SQLIAP	**:** Structural Query Language Injection Attack Prevention
WAF	**:** Web Application Firewall
XML	**:** Extensible Markup Language

List of Figures

List of Tables

Abstract

Cyber security is the study of all aspects of communication security and privacy, and it is dedicated to protecting computer systems from attacks that compromise the hardware, software, or information. A Structured Query Language Injection is one of the most common cyber security attacks on the database of a web application. The attack is a common and dominating type of major web application assault, as well as one of the most serious cyber security threats in which hackers gain access to data. A hacker could simply gain unauthorized access to the web application's underlying database, giving them complete and total control of the system. Many methods and approaches for preventing Structured Query Language Injection Attacks have been developed by several researchers. A deep learning Convolutional Neural Network was used to create a model to prevent Structured Query Language Injection Attacks in this study. In this study, the primary data was collected from Kaggle (SQL injection attack dataset) and it contains a total of 4,199 number SQL injection attacks query and normal text. the data splitting used is 80%-20% for training and testing respectively Furthermore, 90%-10% of data partitioning has experimented. The experiment conducted suggests the 80%-20% data splitting achieved a good result, In addition, the proposed model was built using five different scenarios in the experiment. The scenarios have different parameters and hyperparamter values. Finally, according to the classification metrics report, the proposed model has a 97% accuracy in detecting and preventing Structured Query Injection Attacks while testing with unseen data. Finally, the proposed model produced promising results when tested on an unknown dataset.

Keywords: Convolutional Neural Networks, Structured Query Language Injection Attack, Cyber Security, Structured Query Language

CHAPTER ONE

INTRODUCTION

1.1. Background of the Study

Cybersecurity is just the study of all elements relating to the security and privacy of communications. Cybersecurity is committed to securing computer systems from assaults that may compromise the hardware, software, or information. Permitting unauthorized usage may lead to leaking sensitive information and causing harm or disruption [1]. Thus it requires an action or process, capacity, or state through which information stored within computer systems should be secured and/or defended against harm, unlawful use, alteration, or exploitation [2].

Structured Query Language (SQL) is a program used to organize, manage, and extract information from a computer database. In truth, SQL only works with one form of database, known as a relational database, and it also allows a user to determine the structure and organization of the stored data and also as the connections between the recorded data items. SQL allows a user or application software to obtain and utilize stored data from the database. It also lets a user or an application program modify the database via inserting new data, deleting old data, and changing previously stored data. SQL is a complete language for operating and communicating with a database administration system [3].

Structured Query Language Injection (SQLI) is just a sort of attack in which the attacker obtains access by entering simple SQL code into a web form input box gaining access to alter data as necessary [4]. An assailant can take full use of this weakness to send direct orders to the underlying database of a web application to lose functionality or confidentiality [5].

Structured Query Languages Injection Attacks (SQLIA) is a method through which hackers obtain access to data, as well as a common and dominating type of major web application assault [6]. A hacker may simply acquire unlawful access to the underlying database of the web application, acquiring complete control over the system [6]. Researchers have developed many ways of preventing this issue [7] [8] [9]

However, there is still a research gap regarding SQLIA. For instance the research by Abay. et al. [7] Work didn't address various parameters to analyze SQL injection attacks. Mamdouh Alenezi et al. [8] Work didn't address the overhead rate of detection and the rate of false positives is not also examined. Nanang. [9] Work didn't address, the model that has been created is not evaluated because of the limited dataset. Therefore, in this research, an SQLIA Prevention model to prevent SQLIA using a deep learning algorithm is proposed.

1.2. Motivation

SQLIAs include access to a database without authorization, extraction of information from a database, modification of an existing database, the elevation of user privileges, or the cause of an application malfunction. At Unity University on January 21, 2020, SQLIA happened, and this attack caused a serious problem in the centralized database. The attackers use SQLI to gain access to information kept in the database server and modify database information, access sensitive data, and all student grades, and attendance lists generated by a system for one semester were lost. Therefore, we wanted to develop a model by using deep learning that will prevent SQLIAs.

1.3. Statement of the Problems

Several projects have been completed in recent years to solve or overcome SQLI problems. Musaab Hasan et al. [10], proposed a model to prevent SQLIAs, they have developed a model using a heuristic-based machine learning approach. In this study, they integrated the advantages of dynamic and static analysis with a machine learning approach. It was decided to use a well-studied dataset that contained all possible SQL statements. MATLAB was used to develop the model in a study. They used 23 various machine learning classifications to train the dataset and test it. The top five of the 25 classifiers are then selected going to depend on the outcomes of the actual positive and actual negative rates. After the classification learners completed the training, they checked the precision of each classifier. To get 93.8% they employed the five most effective and accurate classifiers. To improve system performance, they should include non-injected SQL statements in their dataset and investigate and test additional functions. The drawback of this work is small test dataset is used.

In QI LI. et al. [11], proposed a technique based on an adaptable deep forest algorithm for identifying sophisticated SQLI attempts. To concatenate the inputs of each layer, the average of the previous outputs was collected and combined with the raw feature vector. As a result, deep forest structures are more optimal for this study. After that, they devised a strategy referred to as the Adobos-based deep forest model. According to experimental data, the suggested detection model SQLI consists of two steps. Offline classes and online testing were in two phases. A total of ten thousand SQL injection samples were gathered. UNION query, executer SQL instructions, error-based injection, and blind injection were among the features collected from various datasets. The drawback of this work is for large –that scale of data it doesn't perform very well and it has lower computational efficiency and better detection accuracy on fewer samples

The author Krit Kamtuo et al. [12] proposed avoiding server-side SQLI through machine learning. According to this paper, the much more widely known is SQLI security vulnerability in online applications discovered by the US National Security Agency. This framework was created to extract SQL instructions from data and was created to allow getting SQL statements out of a dataset and mark the dataset as an input attribute. Input attributes are passed to the machine learning model to predict SQL injection. This study describes machine learning models, research approaches, and plans. According to the author, variables that receive user-supplied values input are dependent, so there is a risk of SQL injection if hostile SQL statements as user input are mixed with the basic task. By training the model with 1100 samples, the author was able to find out which instructions needed to be avoided and eliminate such weaknesses. This framework needs to improve its ability to detect and predict SQL injection web-based applications by validating SQL syntax. How to detect and block SQLI.

SQLIAs impact many businesses, hence this research work focus on SQLIA prevention. Given the prior research's limitations such that it is not further studied on SQLIAs and examines these attacks depending on many aspects as well as due to the limits of the dataset they have, evaluating the model that has been created is challenging. In this research, we propose robust deep learning or a machine learning technique for preventing SQLIAs. The research work attempt to address the following research questions.

RQ1: How to develop a SQL injection prevention model using deep learning methods?

RQ2: What is the effectiveness of the SQLI prevention model?

1.4. Objective

1.4.1. General Objective

The general objective of this study is to develop an SQLIA prevention model via a deep learning technique

1.4.2. Specific Objectives

To meet the overall purpose of the study, the following specific objectives are developed.

- To conduct a literature review on SQLI detection and prevention methods
- To select an appropriate methodology and tool to build an SQLI prevention model
- To identify deep learning techniques to design prevent SQLI preventing model
- To assess the model’s performance using benchmark datasets

1.5. Methods

1.5.1. Literature Review

Examine further literature on this topic to better understand SQLIA injection attacks and clarify specific solutions to the problem. Criticism of related works will help to obtain the necessary information concerning a specific subject and determine the most appropriate strategy and tools to address the identified challenge. The literature review method concentrates more on the SQLI prevention using various methods such as machine learning which will give us a better understanding and expertise and helps to keep the research going as planned. As a result, books and scientific publications are reviewed more deeply.

1.5.2. Data collection

Our approach is machine learning-based and it necessitates a vast amount of data to finish the task. The SQLIA sample dataset that we are going to utilize was obtained from Kaggle's website [13].

1.5.3. Prototype Development

Certain components are needed to develop a prototype to fill the research gap. Various approaches have been evaluated to build machine learning models that help prevent SQLIAs. Machine learning models that help prevent SQLIAs have been studied using a variety of methods. Python is a computer programming language utilized in the implementation of recommended model algorithms.

1.5.4. Testing and Model Evaluation

The study utilizes assessment metrics, such as accuracy, to assess the effectiveness of the suggested model. The Accuracy metric assesses how well our model works with the data.

1.6. Scope and Limitation

This research focuses on strategies to prevent SQLIA. As previously stated, SQLI is a cyber assault used to obtain control of a database and steal data from it. This study focused on preventing some types of SQLIAs and insider exploitation attacks in a DBMS environment by developing a valid framework. The limitation of this investigation is the fact that dozens of cyber security threats, of which this research paper deals only with SQLIA prevention against databases. Researchers believe that the main limitation is not having sufficient cyber security experience to carry out investigations at the level we should be.

1.7. Significance of the Study

The application of this thesis work could potentially be used to protect all types of data against theft and damage, so companies would not have to worry about unauthorized users accessing their data. Furthermore, it increases the detection and catches rate for threats by stopping them before they reach the database, and also this application contributes to greater safety services in the organizations, as it provides greater security and protects the organizations' data. To sum up, it maintains and supports all the mechanisms that prevent SQL Injection attacks.

1.8. Structure of the Thesis

This document is composed of six sections. Chapter two is about works of literature review and related works this contains a survey of SQLIAs mitigation in many articles, a dissertation, and previous work on this particular topic and then Chapter three presents the proposed model, and its components then Chapter four presents the implementation and experimental result of the proposed system and also describes the overall approach, research design/methods, data collection methods, and procedures, experimentation methodologies, and techniques, validity, and reliability are all discussed in this section. Chapter five focuses on the conclusion of the study according to the findings as well as provides possible recommendations for future SQLI prevention using other approaches and algorithms.

CHAPTER TWO

LITERATURE REVIEWS AND RELATED WORKS

2.1 Introduction

In this chapter, cyber security, SQLIA, how SQLI works, several forms of SQLIAs, protection of multiple kinds of SQLIA, and machine learning techniques including supervised, unsupervised, decision trees, and deep learning, and explain the related works of other scholars on SQLIA will be discussed in depth.

2.2 Cyber Security

A set of procedures aimed at safeguarding a customer's or organization's digital environment and responsible for methodologies used to secure organizations, applications, and information from unapproved access. Shielded from digital dangers by a web-associated framework containing equipment, programming, and information [14]. cyber security is Measures undertaken that secure computer systems from illegal access or attack or actions taken to preserve an individual, organization, or country and their computer information against internet-based crime or cyber-attacks or the state of being safe from illegal or unlawful use of electronic data, or the techniques used it to acquire it [15].

2.3 SQLIA

If the SQL injection assault is effective, it reads sensitive information from the database alters the database information embeds, upgrades, and erases perform database [16]. An SQLIA could be a shape of injection assault in which a preset SQL command is injected into the information plane's input and executed [17].

2.4 How Does SQL Injection Work?

First, SQLI is defined, and then go through how SQLIA works, including the notion and logic behind it, how SQLIA may be used to get information from web applications, and finally, how to avoid SQLIA is also defined. SQLI is the best-known assault and requires a web application that uses a database. For example, consider the web application, which uses a database. This web application may take into account user input as well as put it away in the database, or it may obtain information from a database and display it to the user. What occurs in either situation is the SQL or database query that is generated on the web application and sent to the database, where it is run and relevant data is written by the web application. So, SQLI is that it manipulates database queries to make them do something are not supposed to do, so modify the SQL query, inject a malicious string into it, and then force it to do something it is not supposed to. When the attacker changes a SQL query, and then this malicious query is submitted to the database, where it is run and appropriate results are returned. SQL may be a code infusion component for executing malicious SQL statements take a closer look at how SQLI works with a website that requires a passcode and user name for login. A lot of web applications consistently require logging in first. Examples include Facebook, Gmail, and Twitter. So, to obtain the features as well as functional divisions of the web application, you must first log in to the web application, which normally entails providing the Password and user name.

Because SQLI only works on web applications that use databases, and while it is assumed that the details of usernames and passwords are recorded and stored in the database, the database and the database table containing all of the usernames and passwords are kept in the database. So, after entering the login information, we press the submit or login button, and the information is transferred to the database and cross-checked against the table If none exist, database clients who match that username and password can log in successfully in the database table, we could still problem to successfully log in. However, if such a thing exists a user with that username the password entered is incorrect the login is ineffective. In general, a login is effective if the SQL query yields some values or returns true values. If this SQL query produces false results, the login is invalid.

As a result, SQLIA is a database attack that alters SQL queries so they return true, even if the attacker does not know the username or password. How can this be accomplished, though? Use a logic gate known as an OR logic gate to accomplish this. Let's start with the OR logic. The OR logic functions B. a and b, which accept two inputs and produce one output, are as follows: If both inputs are false, the outcome will then be false. If one’s inputs are correct, the result is genuine. If all is true, the outcome is true of inputs are true. If one’s inputs are genuine, the result will always be genuine, regardless of the other inputs. Last, the yield will continuously be genuine if any inputs are genuine. This SQLIA feature or function has a SQL query, and the goal is for this SQL query to return true. As a result, we'll update the SQL query to something like this.

Table 2.4-1: Sample SQL query

Select * from the user
Where
Username = ‘’ OR 1=1- -’

The primary inverted coma is utilized to close the string parameter at whatever When you type something into a web application's input box, it's when it's handled like a string initial inverted comma is used to close this parameter string parameter in most circumstances, notably when it comes to user names and passwords, and then there's the OR function. Then there's the expression 1 = 1, which includes two inputs, one on the left and one on the right as can be seen. What we're interested in is the statement 1 = 1, which is the right side of the OR function, One is always equal to one, and if one of the OR function's arguments is true, the other input is true as well. This returns true since one is always one and that is true, hence the SQL query always returns true. This sign (‘’) is used to comment on the remaining SQL query, thus the SQLI doesn't matter after that. The SQL query will also return true if the OR function returns true, indicating that the login was successful. SQLI works in this manner.

2.5 SQLIAs Type

SQL-Infusion may be abused under a variety of various methods and can cause major problems. It will also be used by an aggressor to evade acknowledgments and access changes and erase data stored in the database SQL injection. It is also possible to use it to carry out action commands in the computer system in specific situations. This could lead to a more malicious attack on the network behind the firewall [18]. There are multiple kinds of SQLI, which are detailed below.

- **Tautology:** This is a typical injection attack when the query is manipulated to always evaluate to true after execution. They are able to log in as administrators or even completely fictitious individuals after Hackers can gain access by inserting code into a conditional expression [19]. The main purpose of this type of attack is to access the application without providing a valid username or bypassing authentication to extract data.
- **Illegal or Logically Incorrect Queries:** The blunder message produced by a web server contains important information for debugging. The attacker intentionally executes the incorrect queries to find the vulnerable parameters from the error message [20]. The primary goal is to find important information from the database based on the error message or logical error returned [21].
- **Classic or in the band:** a most popular and simplest Classic SQLI is utilized to take advantage of SQLIA. This is brought in-band SQL infusion when an aggressor is capable send off an assault and collecting results over a comparable communications channel [22].
- **Error-based:** In this sort of SQLIA, the attacker creates an incorrect SQL command statement as the attack payload and sends the system anomalous error information. The attack was a successful use of such a system by using this overly suggestive abnormal error information [23].
- **Union Query:** Attackers exploit parameter flaws to change query results This method could be used by an assailant to fool a program into returning data from a table That was not the developer's intention; the developer's primary goal is to use union queries to reveal sensitive data, including configuring the operator [24].
- **Blind SQLI:** An attack known as blind SQL injection involves the attacker posing true-false questions to a database and then determining the response based on the application. This approach is frequently it's when a web app is set up to show generic error messages but the vulnerable code has not been fixed [25]. The main purpose is to ask a series of logical questions about SQL statements and not to forget or hide the error message.
- **Buffer Overflow:** These types are almost the equivalent attack, the main distinction is that the nearby assault is not a network assault. The buffers are

polluted with outside information, causing overflows and being utilized as a chance for pernicious activity [26].

- **Piggy-Backed Query:** it's a kind of attack some extra SQL statements that are malicious are introduced into the database to the unique query at once to corrupt the database. This assault does now no longer alter the authentic question assertion just like the tautology assault, however it locations a further assertion following the SQL query with a semicolon [27]. The purpose of these types of attacks is to alter or steal data, delete or delete facts maliciously, extract data, upload or alter data, carry out denial of provider and execute far-off commands.
- **Stored Procedures:** with this approach stored methods are helpless to assailants who abuse databases. Stored procedures are code that is put inside the database and achieved at once through the database engine [28]. Its main motive is to carry out command execution operations to get entry to the host working machine through an acting denial of service, privilege escalation, and remote commands.
- **Inference**: This category might be utilized for injection-vulnerable parameters and subsequently retrieve data from their schema-identified database. The important reason is to discover injectable parameters, extract facts, and determine database schema [29].
- **Alternate Encodings:** attackers try and hide the inserted textual content and keep away from detection through protective coding strategies and automatic prevention strategies. This sort of assault happens when attackers use alternative encoding, including hexadecimal, ASCII, and Unicode to alter the injection request. The principal reason is to apply opportunity encodings (together with hexadecimal or ASCII) to cover the attacker's sample and keep away from detection [30].
- **Conditional errors:** If the WHERE articulation is correct, it limits or forces the database to evaluate the message that caused the error, resulting in a SQL error [31].
- **Function Call Injection:** You can use program functions to run your web application, process database queries, make internal calls within your web application, and modify your application's data [32].
- **Escape characters with proper filtering:** This attack doesn't stop the user from writing something they don't want to write, so it's passed to the SQL statement.

Finally, it only creates complications for users when they're working with SQL statements [33].

- **Second-Order Injection:** To carry out this attack an insert command plays an important role. To insert the information into the database, the attacker first uses the user's authorization to execute the insert request. The attacker then used the data to control the database [34].
- **Timing Attack:** Time delay is a kind of blind SQL injection. According to the logic inserted, the SQL engine can execute long queues or time-delayed instructions. An attacker can find out by measuring the time it takes for a page to load if the inserted statement is true and an attacker collects information by observing the response time (behavior) of the database [35]. The main purpose is to use the "wait for" keyword to delay the outcome of the database.
- **Conditional response:** Attackers gather information by monitoring the response time (behavior) of the database. Such an injection proves that blind SQLI is possible and an attacker is being allowed to plan a statement that can determine reliability based on the contents looking at a different situation [36].
- **Database server vulnerabilities:** The Vulnerability will exit within the underlying software system, web application, or database system itself. Hackers usually test for SQL entry vulnerabilities by sending application input that causes the server to generate invalid SQL queries. The server then sends a message of failure to the client allowing the attacker to succeed in an SQLIA based on the error, even if user input is hidden [37].

The next stage is to avoid SQLI once grasped the fundamentals of SQLI and the many types of SQLI. SQLIA protection is needed to prevent cybercriminals from extracting sensitive data. For example, credit card numbers, deleting or altering data, deleting tables, stealing credentials, and inserting malicious code. The approach described here shows how to avoid SQLIAs [38].

2.6 SQLIA Prevention

The greatest danger to web managers remains SQL injection assaults, but website owners can take action to reduce the risk [39]. The SQLIAs prevention of methods is listed below.

2.6.1 Static Prevention Techniques

These techniques are very host language-specific, depending on your domain-specific knowledge of how to build queries within the language to identify injection attacks. For SQLIA, it analyses the SQL query statement also on web applications for preventing SQLIA [40]. Static Prevention Techniques are classified as follows.

2.6.1.1 Byte Code Review

This approach tries to target a suspected source of SQLI in the application program's core [41].

2.6.1.2 Parameterized Queries

Parameterized queries are a method of precompiling SQL statements and enable you to clearly define parameters for executing the statements. The database will be able to understand the code and separate it from the input data in this manner. This coding approach helps prevent SQLIAs by automatically quoting user input and ensuring that the input provided does not change the intent. In PHP, data objects are used to store information (PDO). PDO has methods that make it easy to use parameterized SQL queries. It is suitable with a variety of databases, not only MySQL, making your code easier to read and maintain [42]. Parameterized Queries can be classified as follows.

- **Prepared Statements:** Database programmers and database end users have been used to create different database queries to get results for performing tasks. Both perform tasks using simple and dynamic queries. The developer must specify SQL code and pass it as a query parameter in prepared statements and parameterized queries [43].
- **PL/SQL:** PL / SQL SQL syntax templates are used in this computer language. In the template for SQL syntax, there are only two categories of values that can be altered. SQL value placeholders and SQL name placeholders are two types of placeholders. Based on well-defined criteria, these placeholders guide avoiding injection into PL/ SQL. But application developers are all in charge of creating good code [44].

- **SQL Syntax Embedding:** SQL is a guest language, and statements are written in various host languages. Neither does the host language know the guest syntax, which introduces an SQLI vulnerability [45].
- **SQL DOM:** It pushes database connections to a series of classes that are strongly correlated with the database schema, and instead of handling SQL database connections Using JDBC, you can access it., and put those classes into text operations. An attempt to create a SQL statement to use [46].
- **Generic Set of Models:** For preventing injection vulnerabilities in your web application, this is an end-to-end program transformation solution [47].

2.6.1.3 Access Controls Based on Roles

The goal of this Control is for the application developer to ensure that each query is conducted by a role that has the least amount of access to it. As a result, SQL Injection is futile because the query will be incapable of causing any real harm [48].

2.6.1.4 Stored Procedures

SQLI isn't constantly steady in saved procedures. Like maximum saved technique languages, whilst carried out securely, a few conventional saved technique programming functions behave like parameterized queries. SQL parameters are expected to be written down by developers in SQL statements that might be routinely parameterized except if the developer does something unusual. The distinction between an organized assertion and a saved technique is that the SQL code for the saved technique is described and saved inside the database earlier than it's referred to as is with the assistance of an application [49].

2.6.2 Static and Dynamic Methods

2.6.2.1 Anomaly-Based SQLI Detection and Prevention

This strategy sets a typical activity pattern and looks for deviations from it to determine whether it is an assault or intrusion [50].

2.6.2.2 Signature Based

To detect known attacks, requests are matched against known attack patterns or signatures. This is now the most often used type of detection thanks to intrusion detection as well as web application firewall technologies [51].

2.6.2.3 Code Analysis

Code analysis approaches are ways of preventive that generate a static model based on the application code, which may be built or read directly from the source [52].

2.6.2.4 Program Query Language

It‘s a language It gives the coder the ability to express a broad range of applications‘ specific code patterns and When a match is detected, the programmer can additionally define actions to do, such as logging important information or even rectifying an erroneous execution just on time [53].

2.6.3 Dynamic Methods

Dynamic SQLIA prevention methods are explained and categorized as follows.

2.6.3.1 Honey Tokens

Honey tokens may be simply installed to help safeguard a broad range of database systems, and they are especially effective for detecting internal information and privacy violations committed by an employee. Honey tokens are put to use in the detection of harmful elements activities within an organization‘s particular system [54].

2.6.3.2 Web Requests

Scanning HTTP requests is one method of identifying SQL injection before the data is transmitted to a server.

2.6.3.3 All User-Provided Data Is Removed

it’s a technique that evolves escaping all the user's input either through URLs, POST method, or any type of input then it puts these values in the query. This is done so that the DBMS can distinguish between user input and the SQL command itself. So first, it takes the command so the DBMS knows that what it takes is the command itself and all other addition will be the user`s inputs. Because of this, the technique is weak compared to the remainder of the techniques. So it isn't prescribed to be utilized commonly in all applications aside from those applications that do exclude any basic information or when the danger can be ignored [55].

2.6.3.4 Hashes

In the database, hashed versions of login details are maintained, and direct user input is not utilized to build a SQL query unless the hash value matches. If the database only contains safe values, this assures that only safe user input is permitted [56].

2.6.3.5 Allow-List Input Validation

The validation system verifies that the kind of enter that the person has entered is allowed. Input validation guarantees that the information type, length, and layout are correct. Only values that have surpassed the validation system may be used in those instances, enter validating is being used to validate the entries before it's far completed with the aid of querying. Helps save your instructions not to being placed into the entered string. It`s like checking who's knocking earlier than commencing the door [57].

2.6.3.6 Avoid Administrative Privileges Or Follow The User Access Principle

Avoid utilizing the root account while attaching your program to the database. Because an attacker could obtain access to the entire system this should only be done when required by the least privilege principle [58].

2.6.3.7 Web Application Firewall

It's a secure communication protocol by checks packet data levels. More specific information is disclosed by inspecting the data part of packets, which is referred to as the complexity of a packet. The WAF is installed as a running service on the webserver or system it is intended to provide protection, particularly at the application layer. Its basic task is to inspect all incoming HTTP traffic before accepting or rejecting it based on the rules established by the network administrator [59].

2.7 Machine Learning (ML)

ML, it’s a growing field of computing techniques targeted at enhancing human intelligence by learning from their observations. In the modern era of so-called big data, they are considered the engine. ML technologies have proved effective in many fields, including pattern recognition, Spaceship engineering finance entertainment computer vision [60]. The ML model can be categorized into ten categories according to how the algorithm is taught and the availability of results during the training session. These include supervised, unsupervised, semi-supervised, reinforcement, evolutionary, ensemble, artificial neural networks, instance-based, dimension reduction algorithms, and hybrid learning [61]. Some of these paradigms are described below Training can be partitioned into three types Unsupervised Supervised and Semi-supervised [62].

2.7.1 Supervised ML

Within the machine learning field, supervised learning has inspired a vast number of interests. Many supervised learning methods have been taken to the processing and analysis of a broad set of data. Among the most distinguishing features of supervised learning is its ability to use annotated training data. During categorization, labels in these case classes are labeled, and a variety of techniques are used in supervised learning approaches [63]. Labeled data is instead applied to train the network during supervised learning. Architecture like CNN, Residual Network, and LSTM are examples of training types supervised.

2.7.2 Unsupervised ML

Unsupervised learning approaches can be utilized to use allow you to partition data that has never been labeled before. Only unlabeled input features are analyzed by the algorithm, which is designed to uncover hidden structures or relationships in the data. As just a result, unsupervised algorithms are extremely beneficial for tasks involving association and clustering [64]. These Algorithms are a helpful tool for detecting newly unknown patterns of multidimensional data that traditional statistical analysis may not be capable of detecting. When the data category is unknown, this method is appropriate. The data for training is unlabeled and it is a statistics-based learning approach for discovering unlabeled data's hidden structure [65]. Architecture like Autoencoder, Adversarial Networks, RBM, Nodes arranged in hexagonal or rectangular grid training type of unsupervised.

2.7.3 Decision Tree

It's a drawing that appears decisions and their results within the shape of a tree. Graph nodes represent decision rules or circumstances, while graph nodes reflect events or selections. Nodes and branches make up each tree. Each branch indicates a value that the node can expect, and each node represents an attribute inside the group to be classified [66].

2.7.4 Semi-Supervised ML

It is extremely effective in application domains like imaging, information retrieval, and biology, which include a lot of untagged data. Because SSL is a blend of unsupervised and supervised, there are multiple sorts of records labeled and unlabeled records [67]. SSL works by merging information that was labeled and data that has not been labeled SSL has been proposed to solve the disadvantages of supervised learning algorithms that can use untagged data [68].

2.8 Deep Learning

This learning is an element of Machine Learning [69]. It has had notable success in different domains especially the processing of natural language and it is more powerful than normal machine learning approaches in terms of abilities to learn and the ability to exploit datasets for feature extraction. Because of its functionality, utility, and usefulness, It is commonly employed this learning is attracting the attention of numerous scholars. Except for old machine learning, this learning does not require complex feature engineering, and its performance typically improves as the amount of training data grows [70]. There appear to be plenty of deep learning techniques from which to choose and common models are described below.

- **Autoencoder:** is mostly used to process high-dimensional data that is complex. To conduct dimensionality reduction, neural networks that learn features or encoding from a given dataset and that learn the representation in the input data set to reduce dimensionality and rebuild the original data set using an unsupervised technique the learning technique is based on a backpropagation implementation is known as autoencoders [71].
- **Restricted Boltzmann Machine:** It's an unsupervised generative model for extracting patterns from data. It's a fantastic way to learn unsupervised features. In the unsupervised initialization of deep learning techniques, RBMs have also been crucial [72]. The concept of classification isn't a required significant aspect of RBM. This is particularly true for data sets like pictures, videos, and sensor signals. By re-entering the data, these all tend to go unmarked, the RBM must also comprehend the data's inherent building blocks and patterns. RBMs may now be used for more interesting problems in a range of domains, such as image classification, texture creation, and medical image processing, thanks to Computational advances and the introduction of new learning approaches [73].
- **CNN (Convolutional Neural Network):** It's a widely used deep learning approach for solving complicated problems. It gets around the drawbacks of common machine learning techniques. In remote sensing, large data, activity recognition, audio scene, segmentation of MR brain images, picture classification, and object detection, CNN is widely employed, in face detection, speech recognition, vehicle recognition, and many more [74].
- **RNN (Recurrent Neural Network):** It preserves a layer's output and feeds it again into the inputs, which can aid in forecasting the layer's outcome. The layer first is created using the outcome of the weights and features. Once it is generated, the

process of this neural network begins, which means that results from the last time step from one time as an outcome, each individual, neuron conducts computations as if it were a memory cell from one step to the next. Allowing this network to propagate by itself and remembering what information it needed for eventual use in this process are both necessary [75].

- **Neural network with deep learning:** It is a layer that sits between ANN's output and input layers and may simulate a complicated nonlinear relationship. This neural network's additional hidden layers aid in the compilation of features from the lower layer, allowing the neural network to simulate complicated data. Information from either the input is transferred to the output in this process feed-forwarded network [76].
- **Deep Belief Networks:** It's a system for deep learning whereby each pair of adjacent layers is paired as an RBM. A two-layer neural network is used. Nodes on the identical layer are still not connected, while nodes in the other layers are completely connected. The layer of input is being used in the train of the parameters of the connections between the two layers, while an output unit is utilized to make the input RBM will be employed for the subsequent layer [77]. Image recognition, understanding (NLP), retrieval of information, language processing, failure prediction, and other applications employ DBN significantly and this network develops a multilayer neural networks model to analyze the potential features of texts, images, and voice [78].

2.9 Related Works

Deep Learning Architectures depending on the uses and types of neural networks, deep learning architecture is categorized into three categories classes. The first one is Generative Architecture in this architecture there are three different models such as Autoencoder is extensively being used for Facial Expression Recognition, Image Denoising, pattern recognition, Speech processing, Fault Diagnosis, Medical Data Analysis, and Anomaly Detection. Restricted Boltzmann Machines is the second mode this model is extensively being used for Time series forecasting, gradient approximation, and input weight determination. The third model is Recurrent Neural Networks this model is extensively being used for Document image analysis, Handwriting Recognition, Biomedical Image processing, software engineering, and channel estimation. The other architecture is Discriminative Architecture in this architecture there are models called Convolution Neural Network and this model is extensively being used for Data Analysis, facial expression recognition lip-reading, Posture Recognition, Biomedical image processing, and Data Mining. The third architecture is Hybrid Architecture comprises processes that are In most hybrid systems, the generating components are both procedural and discrete and are used along with discriminative components to attain the final solution [79].

SQLIAs have attracted a great deal of attention lately and numerous scholars have researched the area. The purpose of this section is to explain the related works of other scholars on SQLIA.

2.10 Static Approach

The static approach is used to locate much vulnerability. For SQL injection assaults, the static approach detects and prevents SQL injection assaults via way of means of studying SQL query statements on web applications. Numerous studies are mostly based on this strategy, which is described below.

Stephen et al. [80] propose an automatic fix generation solution that replaces vulnerable statement-based SQL statements with protected prepared statement-based SQL statements. They use a conversion algorithm to preserve the logic of the statement while saving the statement without knowing the context. This is the reasoning behind this solution is that it can automatically generate fixes to protect vulnerable SQL statements without the need for software security expertise. The limitation of this work is that they can use conversion algorithms to convert large numbers of vulnerable SQL statements, but not batch SQL statements. The JDBC prepared statement interface does not currently allow multiple independent queries on the same batch prepared statement, so it cannot process batch jobs. And another limitation is that this work was originally intended for Java, and the concepts and general algorithms could be used for other languages.

Zeinab et al. [81] proposed a tool for SQLIA to report the transformations that need to be performed to protect the underlying database and web application. They used static analysis, based on the results of the analysis, and also used new detection techniques to generate the required transformations and report them as output. Discovery technology removes user input for SQL queries and collects some information to make run-time discovery easier and faster. Finally, they delivered the evaluation's findings. The downside of this task is that you can manually perform the conversion to increase the security of your web application, but this process can be tedious and needs to be automated.

Bart et al. [82], recommended the creation of a static analysis tool to handle these web application input-related concerns. When utilized in the framework, they employ an abstract model of the source program that takes user input and dynamically produces SQL queries. The framework employs novel checking methods on these state machines to demonstrate or validate that the original application program is secure. The analysis is already in the works and being tested. This work limitation is that using the FSA under-approximation sub-approach to the SQL grammar can be too restrictive to remove malicious queries from the presented set, and some operators like "LIKE". Is still unlikely to handle. 'And' ×'. 'The generated constants cause similar problems in automaton-based analysis. Finally, to experimentally check the tool's efficiency as a means of analysis, there is no practical prototype of the analysis and it has not been applied to the actual examples.

Xiang et al. [83], proposed SAFELY, a technique of static analysis that can automatically produce test cases that exploit SQLI vulnerabilities in ASP.NET online applications as a crucial design proposal offered. The approach for calculating/estimating the satisfiability of string constraints is what makes this tool unique. By symbolically running an ASP.NET web application, SAFELY generates string equations that match a specific attack pattern. SAFELY, when completely built, will be capable of using source code information to find extremely sensitive vulnerabilities that Vulnerability scanners that operate in the background are known as black-box scanners Are unable to detect. The implementation of SAFELY and the investigation of techniques for automatically enumerating SQL WHERE clauses, when they are not closed, are the work's limitations.

2.11 Dynamic Approach

Dynamic analysis is another approach used for detecting SQLIAs, it prevents vulnerabilities during program execution. Numerous studies have presented solutions based on this method.

Dennis et al. [84], give an outline of a different way of identifying SQLIA Dynamic Analysis, which detects vulnerabilities while the program is running. Introduced 4SQLi, an automated technique based on a collection of mutation operators that change inputs to create new test inputs to trigger SQLIAs, to detect SQLIA based on dynamic analysis. This could result in inputs including new attack patterns, boosting the chances of finding vulnerabilities.

2.12 Hybrid Approach

Combined Dynamic and Static Analysis would be a technique for detecting and preventing SQLIAs that combines the benefits of both techniques. Numerous studies have presented solutions based on this method.

William et al. [85] Presented a solution named AMNESIA, which stands for analysis and monitoring, to neutralize SQLIA in AMNESIA detects and prevents web application vulnerabilities in real-time by combining dynamic and static analytics. AMNESIA creates several sorts of queries using static analysis. AMNESIA understands all queries before they are submitted to the database and tests each query against a statically created form in the dynamic segment. The findings of this study indicate that AMNESIA is a very useful and potent method for detecting and preventing SQLIA. In some cases, the technique described in this article could result in false negatives. False positives can happen when text parsing isn't precise enough, and false positives can also include malicious queries in the built-in SQL query model, which could be one of an attacker's malicious searches. If you can create a matched injection attack, this can happen.

Ahmad Ghafarian [86] proposed a new method to determine how else to resist SQLIA. This method is a combination of dynamic and static strategies. The proposed approach suggested enlarging all database tables in the main phase (static) to include information that only includes a few symbols, including the greenback symbol. This must be done throughout the stage of database design before implementation. The limitation of this paper is It isn't always implemented, the set of rules may be extended, and it does now no longer encompasses different varieties of SQLIA.

Inyong et al. [87], proposed an original strategy for identifying SQL infusion assaults by contrasting static SQL inquiries and powerfully produced questions after eliminating the property estimations. Besides, we assessed the presence of the proposed technique by probing weak web applications. We additionally contrasted our technique and other discovery strategies and showed the effectiveness of our proposed strategy. The proposed strategy just eliminates the property estimations in SQL inquiries for examination, which makes it autonomous of the DBMS. Complex tasks, for example, parse trees or specific libraries are not required in the proposed strategy. The proposed strategy can't be done on web applications yet it can likewise be utilized on any applications associated with data sets. Besides, it tends to be utilized for SQL inquiry profiling, SQL question posting, and modularization of location programs. The limitation of this work is required for SQL infusion assaults as well as for other web application assaults like XSS, in light of the proposed technique and AI algorithms.

Raymond et al. [88], proposed an ASSIST method for programmed query sanitization. This utilizes a mix of static investigation and program change to consequently find and sanitize the factors used to create SQL queries. They executed the technology usimg Java ptogramming language using a device called ASSIST to protect Java byte code (got from JSP or Servlet). An experiment has revealed that help works well for a suite of SQL infusion assault tests, and helps work with little run-time overhead.

However, the limitations of this work incorporate direct comparisons with other techniques, calculation changes to lessen potential inaccuracies, and techniques for exploring other user-input properties (such as assignment strategies and allotment techniques). Incorporates a more complete assessment, like expansion. Many theoretical questions can be over-approximated because the comparison is not performed appropriately.

Ramya et al. [89] proposed a structure called the runtime checking framework. It is utilized in procedures for creating run-time screens that perform run-time monitoring of web applications after deployment to recognize and forestall tautology-based SQLIA.

With our structure, application quality and security are accomplished in the pre-arrangement stage, yet additionally in the post-sending stage, and the potential abuse of vulnerabilities by outside attackers is identified and forestalled. Additionally, the authors explained the assessment of the proposed method, and the outcomes acquired showed that their method can handle all tautology-based SQLIA successfully and that real input can access the information base. They will additionally utilize the proposed framework to computerize the whole course of developing a runtime monitor and stretch out the system to recognize and forestall any remaining kinds of SQLIA.

Wang et al. [90] Depending on the kind of SQLIA, they have introduced a new way to identify and forestall SQL injection assaults utilizing AOP. From one viewpoint, it resolves these SQLIAs with assault properties by characterizing aspects and pointcuts and doing some validation with the Before function. In other attacks, and also a model-based course that uses programmatic research techniques to consistently assemble a model of a real SQL query and compare that model to a SQL query that has been significantly rebuilt from AOP. They also use contextual analysis on the simple client login page as the most effective method for accomplishing this as an aftereffect of this test, this technique can forestall web applications from all SQLIA. However, the limitation of this task is that it needs to be analyzed the source code.

Emon et al. [91] The authors developed a model which is a web-based multitier architecture. The proposed model prevents various SQLI. The model shows its productivity by tracking various SQL injections, and the performance results show its applicability. The positive rate for most types of SQL infusion attacks is 100%. The limitation of the model is that it is very vulnerable to inline SQL injection attacks and requires a different approach to reduce execution time, which is highly susceptible to SQL injection attacks.

Table 2-1: Summary of SQLIA prevention using ML and DL

Paper	Objective/Purpose	Methodology	Research Gap
Kevin et al. [92]	Established that the algorithms experimented with, such as rule-based and decision tree algorithms, have achieved accuracy close to that of Neural Networks in their experiments and are much good in terms of time required to build models and execution time when classifying testing data, and have established that the algorithms.	The data generating strategy involves three stages: traffic production, capture, and preprocessing. Rule-based, Support Vector Machine, Neural Network, and Random Forest algorithms are employed.	Extra data collecting, such as outbound traffic from the web application to the browser, Larger datasets should be collected to determine if this improves performance, as well as analysis of additional machine learning techniques for accuracy and performance, and adaptation of this system to identify other sorts of web-based assaults.
Auninda et al. [93]	Developing a method for successfully detecting and preventing SQLIAs, as well as developing a model by determining the optimum machine learning algorithm for predicting and preventing SQLIAs Attacks. This document offers a summary of the work plan, experimentation, and the results of the experiments.	5 different methods were used to train a sample dataset to see how accurate and employed Logistic Regression, Neural Network Classifier, Random Forest, KNN, and Naive Bayes as algorithms.	The model proposed in this research uses an ML technique to learn about new types of SQLIAs and recognize them in the future.
Sonali et al. [94]	The study's overall goal is to employ a Gradient Boosting Classifier method	To categorize and detect SQLIAs, a gradient boosting method is used.	There is insufficient data to train machine learning models and refine them in

	from ensemble machine learning approaches to classify and detect SQL Injection threats in their research. Gradient appears to be A suitable learning strategy for reducing errors and providing more accurate predictions. To give results, the Gradient Boosting technique uses simple classifiers, primarily decision trees, in a sequential way.		terms of usability and efficiency.
Ines et al. [95]	The following are the paper's main contributions: 1. The SQLI attack was described in detail. The various sources, goals, and forms of attacks are defined and discussed. 2) A classification of the different SQLI attack detection and prevention countermeasures are presented and discussed. 3) A table evaluating the various potential SQLI attack countermeasures was published. 4) Newly proposed solutions are outlined	different existing countermeasures that were proposed to either detect or prevent the SQLI attack like A. Query-model based SQLI countermeasures, Obfuscation based SQLI countermeasures, Monitoring and Auditing Based SQLI countermeasures, Entropy-Based SQLI countermeasures, Ontology-Based SQLI countermeasures, Machine Learning Based SQLIA countermeasures	There are still flaws in their ability to deal with web-based threats. And creating a machine-learning-based SQLI attack detection technique that can handle a huge number of queries per second. Also, there's a shortage of an up-to-date and consistent dataset that researchers may use to evaluate their work and compare it to existing solutions

	and discussed, like ontology		
Latchoumi et al. [96]	The proposed approach model could have been used to report vulnerabilities in online applications. As a result, this method may reduce your web application's odds of launching SQLIA. Machine learning with SVM algorithms is used to avoid SQLIA runtime monitoring.	To avoid SQLIA runtime monitoring, ML with SVM methods is applied. When the home page of each application is transferred to a test page, the answer behind this strategy is to detect and avoid SQLIA outages.	The methodology provided here only works in a data-rich situation, which is insufficient for this research.
Nanang et al. [97]	The design of the recommended approach for how the SQL injection detector works using deep learning and transfer learning methodology is explained, as well as the implementation and results of the SQLi detector using AI. In this approach, transfer learning techniques are used to produce detectors.	The SWIVEL architecture, which is one of TensorFlow's neural network models for text embedding, is used to create the system.	The drawback of this effort is that due to the dataset's limitations, evaluating the model that has been constructed is difficult.
Stanislva et al. [98]	SQLI and XSS are two of the major code injection attacks in today's web applications and systems.	CODDLE, a DL-based IDSS for web-based assaults is proposed in this work, along with numerical experiments on datasets for both SQL and XSS attacks. CODDLE's type encoding improves the data rate from around 75% to 95% accuracy, 99.5% precision, and 92.5% recall.	This paper does not employ a pre-processing function, instead of converting each query symbol to a char code. This sort of encoding is vulnerable to attacks that alter the symbol sequences in the injection

			query.
Ding et al. [99]	In this study, we offer a non-background rule-based SQLi detection method based on an NLP model and deep learning framework, as well as a lightweight solution to SQLIA prevention based on conditional embedding and CNN and MLP.	This research develops an SQLI detection system using a DL architecture and lexical analysis approaches. The experiments were compared using a CNN and MLP.	This research only focuses on only first-order SQLI. Second-order injection and hybrid attacks were not researched.
Kevin Zhang [100]	To create a machine learning-based classifier that can detect SQLI vulnerabilities in files. It also compares the effectiveness of traditional ML algorithms and deep learning-based approaches for detecting SQLI.	Utilizing input data validation and sanitization aspects, machine learning was used to train and evaluate classifier models. The highest precision (95.4%) was achieved by a CNN, while the highest recall was achieved by a Multilayer Perceptron (MLP) model (63.7 %).	Word2vec model did not perform well, and currently missing a word embedding model designed for PHP source code. And it is only limited to the PHP language.
Maruf et al. [101]	To develop a model for identifying SQLI vulnerabilities in a web application using deep learning by extracting various web application vulnerability finding points.	The study compares different classifiers used in this study to see if any other machine learning algorithm can improve the neural network's results. An SVM, random forest, and Naive Bayes with an accuracy of 94.66 %, 97.33 %, and 84.49 %, respectively, were clearly shown.	SVM has the disadvantages of being time and memory-consuming, difficult to choose the right kernel function, high algorithmic complexity, and

			inability to work with large data sets..
Ao Luo et al. [102]	To see if the CNN algorithm can be used to detect SQLIAs and compare it to a traditional detection method called Mod Security.	SQL injection detection is based on CNN and Mod Security, a web application intrusion detection and prevention engine that can also act as a web application firewall (WAF).	The paper's weaknesses are primarily focused on the HTTP log file, rather than the proper dataset to train on, so it can be said that the dataset's quality has an impact on the deep learning model's performance.
Manav et al. [103]	To investigate the numerous techniques for trying to identify and protect SQLIAs. The performance of five different classification models was compared	Machine learning algorithms such as Naive Bayes, Decision trees, Support Vector Machines, and K-nearest neighbors have been tested in all major types of SQL attacks. And experiments show that CNN outshines other algorithms with 94.84 present accuracy, 85.67 present precision, and 96.56 present recall.	It does not apply feature reduction techniques to study its impact on performance measures, and it does not expand the dataset by adding other types of SQL injection attacks as and when more types of queries arise in the future.
Jothi et al. [104]	It would be able to sense any and all injection techniques. The model will handle all of the feature	Researchers accomplished a cross-validated accuracy of 98 % with a precision of 98 % and a recall of 97 % using the MLP model.	The model only uses single-word tokenization and excludes N-gram models. The spatial and temporal features

	extraction and selection. The client will only have to type the text.		of SQLI are not learned using CNN-based architecture.
Tonmoy [105]	A DL framework for sensing SQLI weaknesses in web services and making a tool that can tell you which web applications are vulnerable to SQLI and which aren't.	A unique data set is generated, followed by data collection, pre-processing, and feature selection, all of which were then fed into a deep learning model. A tool has been developed to detect SQLI vulnerabilities automatically.	This work does not dig into deeper a web application to invent a more efficient way to detect the attack and find an efficient solution to prevent that attack
Jiabao et al. [106]	To identify web attacks in the HTTP request protocol, a CNN and LSTM network architecture was introduced. SQLI, XSS, and other script injection attacks are examples of anomalous requests to be detected.	This study compares the combined performance of LSTM and CNN with other traditional methods such as Multinomial Naive Bayes (NB), Linear Support Vector Machine (Linear SVM), Neural Network (NN), k-Nearest Neighbour (kNN), and Decision Tree (DT) and with 0.989 %Precision and 0.988 % recall	The CNN and LSTM methods are resistant to varying dropout rates. Moreover, when the rate is greater than 0.3, performance increases slowly.
Huafeng et al. [107]	This paper's purpose is to find SQLI attacks in network traffic using deep learning.	It uses deep learning to train the sample data in order to detect SQL injection attacks, and researchers compared different algorithms such as LSTM, MLP, CNN, and DBN to find that the DBN model	More data is needed in this study to improve the model's accuracy.

		had the best accuracy rate of 0.9603.	
Ming et al. [108]	Using just a specially built CNN, a DL approach for detecting Web attacks.	The experiments shown on the dataset HTTP DATASET show that the designed CNN performs well and achieves satisfactory results in detecting Web attacks, with an accuracy of 96.49%.	It only focused on detecting the server-side attacks only. And the method in this article only focuses on detecting Web attacks hidden in URLs
Yong et al. [109]	SQL query strings first were syntactically evaluated into the tokens, and then the likelihood ratio test was used to construct a word vector of SQL tokens, before training an LSTM model with sequences of token word vectors.	A tool called WOVSQLI describes and develops SQLIA based on the LSTM and word2vec of SQL tokens neural networks. Experiment results show that WOVSQLI can easily detect SQLIA with 98% accuracy.	Future work will improve the model with a new dataset

CHAPTER THREE

THE PROPOSED SQLIP MODEL

3.1 Introduction

This chapter focuses on the design of the proposed model and its experimental setups. To mention the contents on a higher level, it contains the system architecture, data sources, and how the data is collected and processed. Following that the proposed model design and detailed descriptions of the features, how the feature is extracted, and how the prevention of SQL injection attack is performed in the proposed deep learning model.

3.2 System Architecture

The system architecture begins with data collection and then encodes and converts the data to CSV file format. Following that the dataset is split into training, validation, and testing set. The training and validation set follow a data preprocessing and selecting the best features are part of the data preparation phase. Following that the preprocessed dataset is fed to the CNN algorithm and it extracts featured during the training phase. Based on the performance of the training and validation set the model evaluation technique was applied. After experimenting with various hyper-parameters, the model with the best performance will be chosen. Finally, the best model will be tested with unseen data using the testing phase. The system architecture of the study is depicted in Figure 3-1.

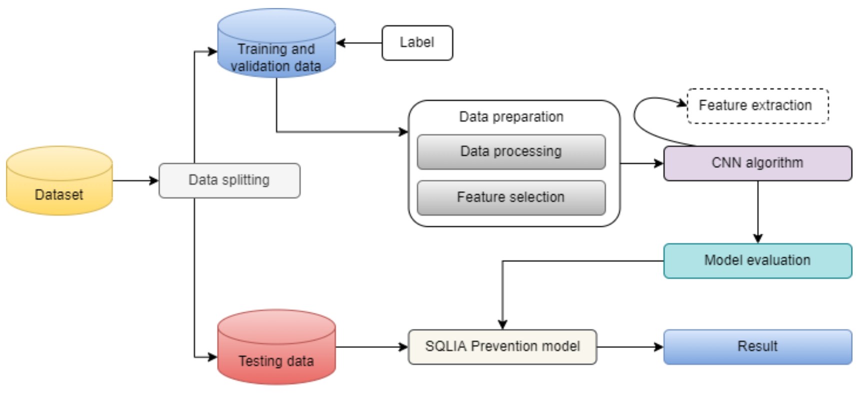


Figure 3.2-1: System architecture

3.3 Dataset Preparation

The goal of data preparation is to use the data as it exists and then clean up what has been done, the way to proceed is to. Data preparation refers to the process of removing outliers and creating a more uniform distribution. It also includes cleaning and standardizing data. Cleansing is done by replacing missing values with their averages or other relevant values, which is one of the most important steps in any machine learning or other research.

One of the most important steps in any machine learning task is understanding the problem domain and the data that will be used in the process. The data preparation process we have followed in the following sections includes activities like data collection, data and business understanding, dataset description, and data formatting. Next, feature selection and the data pre-processing stage are followed by applying data cleaning.

3.3.1 Data Collection

In this study, the primary data was collected from Kaggle (SQL injection attack dataset). The data was prepared manually, and it contains a total of 4,199 number SQL injection attacks query and normal text. The dataset has been prepared by collecting different SQL injection queries, and it has been labeled as class 'SQLI ' and 'NORMAL'. The statistics of the dataset and the sample data of each class are depicted in Figure 3-2 and Figure 3-3 respectively.

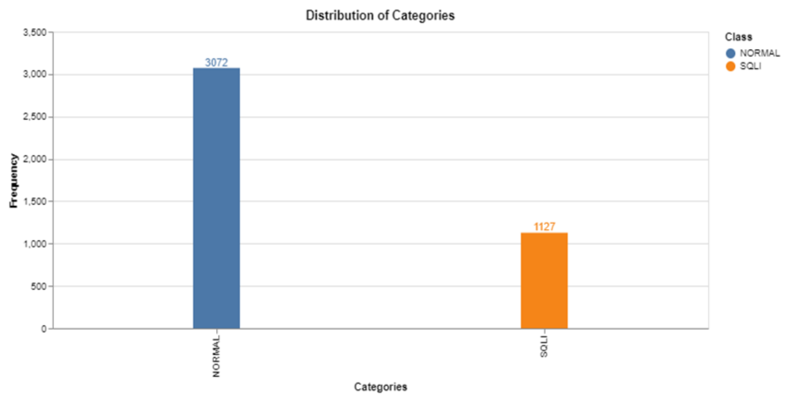
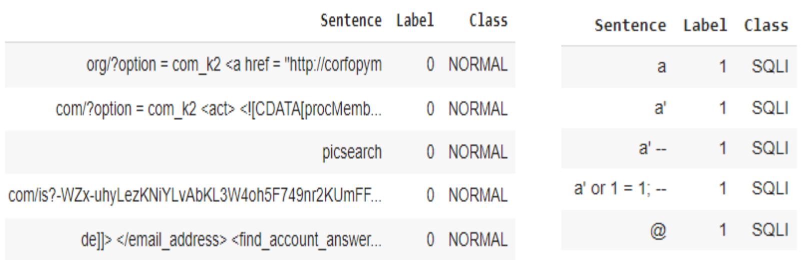


Figure 3.3-1: Distribution of dataset



Sentence	Label	Class
org/?option = com_k2 <a href = "http://corfopym	0	NORMAL
com/?option = com_k2 <act> <![CDATA[procMemb...	0	NORMAL
picsearch	0	NORMAL
com/is?-WZx-uhyLezKNiYLvAbKL3W4oh5F749nr2KUmFF...	0	NORMAL
de]]> </email_address> <find_account_answer...	0	NORMAL

Sentence	Label	Class
a	1	SQLI
a'	1	SQLI
a' --	1	SQLI
a' or 1 = 1; --	1	SQLI
@	1	SQLI

Figure 3.3-2: Sample dataset

3.3.2 Understanding SQLI Data

Before taking any steps, the main activity in any machine learning task is data understanding. It begins with gaining initial insights into the data and the problem domain. Later, it proceeds on to other tasks such as evaluating the quality of the data, missing values, and outliers that could have an impact on the machine learning final result. Understanding the data aids in identifying useful and interesting insights into the data that will be used to develop a hypothesis for the hidden or unknown information.

3.3.3 Business Process Understanding

Understanding the overall process and procedures that are undertaken in the problem domain is at the heart of what business understanding is all about. This can be obtained through a variety of methods, including observations, document analysis, and discussions with domain experts. Understanding the problem domain aids in gaining a better understanding of the data that will be handled, as it provides insight into what is happening and why. In this study, problem domain data is gathered from open-source dataset sources derived from various SQL injection sources. In general, the dataset contains two types of data categories: SQL injection queries and normal text.

3.3.4 Data Formatting

The dataset was prepared by collecting different SQL injection attacks, which were collected from different websites. The collected SQL attacks were merged and preprocessed to train using different machine learning and deep learning algorithms. Finally, the merged Microsoft Excel file was converted to CSV (Comma Separated Values), which was then used for training and testing.

3.4 Algorithm Selection

Deep learning makes things easier than conventional machine learning algorithms because it makes feature extraction and other preprocessing tasks automatic. In this study, CNN algorithms selected deep learning to build a proposed model, which is best for text-based classification and other problems. Several works of literature claim that CNN is best for classification and detection problems, which yields the best result and it is a state-of-the-art performance having enough data and optimal hyper-parameters. In summary, a convolutional neural network model has the potential to automatically prevent SQL injection attacks.

In this study, the proposed model was built from a model from the scratch without applying different CNN architecture types like XLNet, ERNIE, and Text-to-Text Transfer Transformer. Additionally, the components, parameters, and hyperparameters of the proposed model are explained in the coming sections.

3.5 Feature Selection

Feature selection is an essential iterative process for selecting the best features or removing variables that do not assist the model in mapping the relationship between the data variables and the target outcomes. To accurately identify the best parameters that represent the whole problem and achieve the objective of the study, the best features must be identified. The expected outcome of the algorithms, based on the problem, is fully dependent on the quality of the selected input features. Reducing highly unrelated features will increase the amount of time it takes to train the model, improve performance, and minimize the complexity of the algorithm [110].

After identifying the best parameters and attributes of the SQLI data, the more relevant features were selected to build a model that would prevent SQLI from using a deep learning approach. Moreover, features that are not relevant to this research objective were removed from the collected SQLI data to learn and map the features or train the machine learning algorithms based on the dataset prepared and labeled by open source.

3.6 Data Splitting

The data collected were classified into three categories: training, validation, and testing. Training, which is used to train the model, and test, which is used to test the model that was not discovered during training. The validation set is used to evaluate the performance of the model created during training and is frequently used to fine-tune model parameters to achieve the best model performance. Then, the test set is used to test the proposed model with unseen data. In this study, the data splitting used is 80%-20% for training and testing respectively. Furthermore, 90%-10% of data partitioning has experimented. The experiment conducted suggests the 80%-20% data splitting achieved a good result.

3.7 Feature Extraction

The term feature extraction refers to the process of transforming raw data into numerical features and processing the resulting features while preserving the information in the original data set [111]. It yields better results than applying machine learning directly to the raw data. The main advantage of deep learning over traditional machine learning is the ability of the feature extraction stage to be done automatically by the CNN algorithm without the need for domain experts. Since deep learning has the potential to derive complex characteristics, it has a significant advantage over traditional machine learning, which has a limited ability to learn [112].

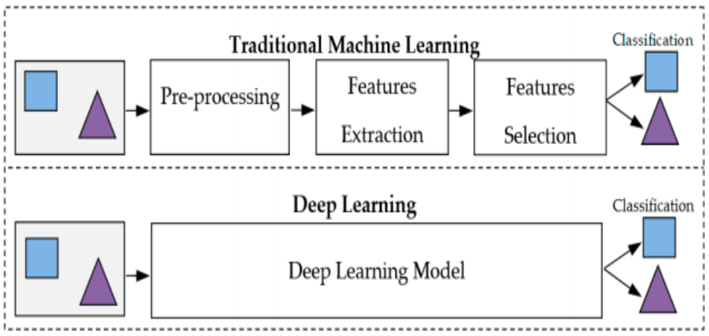


Figure 3.3-3: Feature extraction ML and DL

3.8 SQLIA Prevention via Classification

The proposed SQLIA deep learning model has three convolutional layers, and three pooling layers (MaxPooling), except the final layer of the CNN architecture the activation function used, is ‘relu’. The final layer used the ‘Sigmoid’ activation function. Based on the components of the proposed model, the SQLIA was successfully prevented by identifying the given query as ‘Normal’ or ‘SQLIA’. Figure 3-5 presents the proposed model architecture.

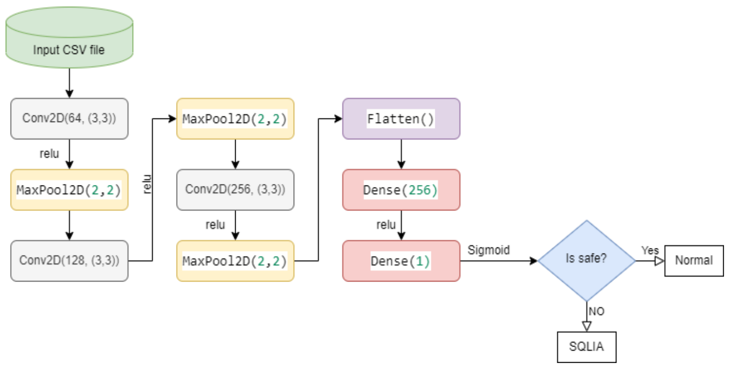


Figure 3-4: The Proposed SQLIAP model

3.9 The Proposed Model Components

The proposed model has different components of the CNN algorithm. The SQLIA model was built depending on the components. Here, the descriptions of the components of the proposed model are explained. The components and the total number of parameters used in the proposed model are depicted in Figure 4-4.

3.9.1 Convolutional Layer

The convolutional layer is one of the main blocks of Convolutional neural networks which is composed of mathematical operations and convolution is a special form of linear operation. Pixel values that are processed in (2d) two-dimensional and grid (number arrays and a narrow grid) parameters are represented by mammogram images or usually, digital images, called the kernel, and an optimizable function extractor is implemented at the image location, making the processing of images extremely effective by CNN. As the layers feed their production to the next layers, the features that are extracted are hierarchically and increasingly complex. To refine the parameters, optimization algorithms such as the backpropagation and gradient descent algorithm are used and the kernel is carried out to minimize the gap between the outputs and the ground truth labels [113].

3.9.2 Pooling Layer

This layer enables for downsampling operation that enables a reduction in the number of learning parameters and several output channels are the same as the number of input channels [114]. This layer enables the extraction of main features in some spatial locations, to manage and control overfitting in the network. The filter size, stride, and padding are hyperparameters in pooling operations, and there are no learnable parameters in pooling layers. Instead, pooling operators are deterministic, usually measuring the items in the pooling window by either the maximum or the average value, these operations are called maximum pooling (max pooling for short) and average pooling, respectively [115].

Max pooling is the most widely used pooling operation with a filter size of P x Q and with strides to downsample the dimension of feature maps. The depth of the feature map is not changed but the height and the width are changed and Max Pooling takes the maximum value out of each sub matrix of a activation map and differentiates it into matrices [116].

3.9.3 Fully Connected Layers

This is the classification layer that calculates the score of each class of the extracted Features of a convolutional layer in the next steps and the last layer feature maps are represented as vectors with scalar values that are passed to the fully linked layers and It guarantees that any neuron within previous layers is linked to each subsequent layer [117].

CHAPTER FOUR

IMPLEMENTATION & EVALUATION

4 Introduction

This chapter discusses experiments to design and develop the SQLI prevention model and evaluate the proposed model via evaluation metrics. In addition, the experimental scenarios were explored using different parameters and hyper-parameter to build the proposed model.

4.1 Tools and Experiment Setup

There are currently many deep learning frameworks available. Among those deep learning, frameworks are TensorFlow, Keras, and PyTorch, which are the most widely used ones. These three are favored by data scientists and deep learning beginners. There is no hard and fast rule on which deep learning frameworks should be chosen, but considering some of the conditions for selecting a framework would be beneficial and appropriate. Here some of the deep learning frameworks are explained.

Keras: Is a Python API that can be used with CNTK, TensorFlow, or Theano is available. Keras assists machine learning researchers in rapidly turning an idea into a product. In this study, the Keras API was used on top of TensorFlow. The following are some of the API's advantages.

- Easier to use and extensible
- Support both CPU and GPU
- Support CNN and RCNN

TensorFlow: This is a well-known machine learning and data science framework. Many libraries, tools, and resources are available through the API, making it easier for researchers to develop and deploy machine learning applications. Furthermore, the API's simplicity makes it simple for developers to train and build a model.

Pytorch: Is a Torch-based open-source machine learning platform for Python. PyTorch was developed by Facebook's AI team and is used for natural language processing and a variety of machine learning projects.

Google Collaboratory: This is a Google cloud tool that allows users to write and run programs or texts in their browser without needing to configure anything. The tool has unrestricted GPU and TPU access. Furthermore, the Google Colab implementation is simple to share across multiple platforms. "Colab Notebook" is a Google collaboration tool that allows you to write code and text in an interactive environment. In this study, the Google Colab was configured with a GPU hardware accelerator and the notebook server was hosted within the environment. In summary, Colab's Notebook looks like the famous Jupyter Notebook with an additional bonus feature which has:

- Access CPU and GPU
- Stored data in a Google drive
- Has a huge library for machine learning, deep learning, data science, and other fields
- Easy to share with other colleagues
- The python code used by Colab can execute anywhere

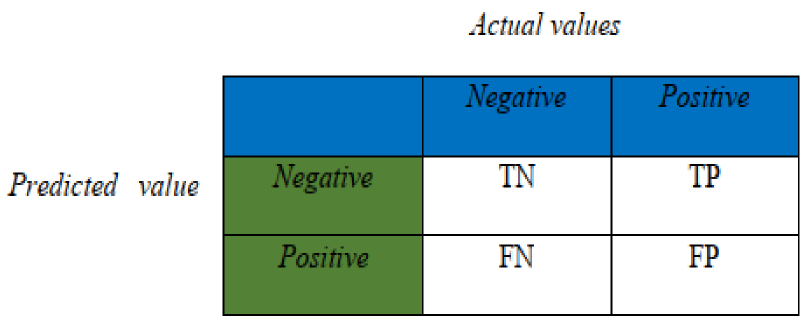
draw.io: This is an online tool for drawing flow charts, and it provides a variety of symbols and templates that enable users to create usage case flow charts, activity diagrams, network diagrams, floor plans, charts, mind maps, infographics, and several. Furthermore, this tool is quite simple, and it can export the final diagram in PNG, JPEG, PDF, and other formats.

Mendeley: is a desktop application that serves as a PDF reader while also allowing users to cite documents in the form of IEEE, APA, and other formats. The tool automatically loads all citation information that you provide, unlike Microsoft Word, and this is much faster than having to type it in yourself.

Figure 4-1: Software tools and implementations

4.2 Evaluation Techniques

After the model is trained with a given set of data, model evaluation is the technique used to determine the degree of accuracy and efficiency of the model. A computational problem, such as classification and detection, uses evaluation metrics such as accuracy, precision, recall, and F1-score to predict which class instance belongs to which class. Those metrics were computed using classification metrics and provided information about a model's results in each class. All of the metrics listed above are determined based on the confusion matrix value, i.e. TP (True Positive), TN (True Negative), FP (False Positive), FN (False Negative).



		Actual values	
		Negative	Positive
Predicted value	Negative	TN	TP
	Positive	FN	FP

Figure 4-2: Evaluation metrics

Where:

TN: stands for True Negative which shows the number of negative examples classified accurately
TP: indicates the number of positive examples classified accurately
FP: shows False Positive value, i.e., the number of actual negative examples classified as positive
FN: False Negative value which is the number of actual positive examples classified as negative

i. **Accuracy**: Answers the question about how often the model predicts the classes correctly i.e. SQLI and NORMAL.

Equation 1: Accuracy metrics

$$\textbf{Accuracy} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+FP+TN+FN}}$$

ii. **Precision**: It gives insight into how often a positive value prediction is correct. Example: Predicting SQLI, how often the prediction precisely predicts.

Equation 2: Precision metrics

$$\textbf{Precision}=\frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}}$$

iii. **Recall**: Also known as sensitivity it describes how sensitive the classifier is while detecting positive instances.

Equation 3: Recall metrics

$$\textbf{Recall}=\frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}}$$

iv. **F1-Score**: This is the harmonic mean of the precision and recall, and the lowest value of the F1-score is 0.

Equation 4: F1-score metrics

$$\textbf{F1-Score}=\frac{2}{\frac{1}{Recall}+\frac{1}{Precison}}$$

4.3 Hyper parameter Selection & Experimental Scenarios

4.3.1 Hyper parameter Selection

Hyperparameters are the variables that are used to build a model in a neural network. An optimal hyperparameter value must be configured before building a robust deep learning model. The optimization algorithm, learning rate, loss function, number of epochs, and batch size are the CNN algorithm‘s hyperparameters. There is no set rule for configuring hyperparameters of a given model, and different configurations can be used depending on the computational problem. Because it requires multiple experiments and there is no general formula for setting good hyperparameters, hyperparameter tuning is a time-consuming task [119]. In this study, the hyperparameter value was determined through a series of experiments. Below is the definition of some hyperparameters and their value is mentioned. Finally, the summary of hyperparameter values is summarized in Table 5-1.

Optimization algorithm: it allows the model to learn faster and perform better. There are a variety of optimization methods, including *Adam* and *RMSprop*, which were tested at learning rates of 0.01. When training a neural network, choosing the correct optimization algorithm is crucial.

The Learning Rate: the value of a neural network is also a tunable hyperparameter. This value of a neural network is used to train a neural network with a value between 0.0 and 1.0 is usually specified.

A slow learning rate necessitates multiple training epochs and weight changes, whereas a fast learning rate necessitates a rapid change in the training epoch. Choosing a learning rate that is neither too high nor too low is one of the most difficult aspects of creating a neural network model. After several rounds of testing, a learning rate of 0.01 was chosen to build the proposed model.

Loss Function: A loss function, also known as a cost function. Let y denote the actual output, $\hat{y}$=predicted output, and K= number of classes. Then, $y - \hat{y}$ was calculated utilizing a cost function called cross-entropy (CE). Binary cross-entropy is used for binary classification problems. In this study, a binary CE was used to build the proposed model because two classes exist.

Number of Epochs: Count how many times the neural network has been exposed to training data. According to the proposed model, the optimal number of epochs discovered through experimentation is 10.

Batch Size: The batch size refers to the number of subsamples sent to the network for parameter updates. Batch size is set to 32, 64, 128, and so on by default. Batch sizes of 32 and 64 were used for experimentation.

Activation Function: This is a function responsible to fire or not firing a neuron based on some inputs. The proposed model uses a nonlinear activation function termed ReLU after each convolution layer. The ReLU activation function is applied in a variety of machine learning problems since it does not suffer from the vanishing gradient problem and leads to faster computations [105]. Additionally, Sigmoid and Softmax activation functions have been experimented with for the last layer activation function.

Table 4-1: Hyperparamter value summary

Hyper parameters	**Value**
Optimization algorithm	Adam
Learning rate	0.01
Activation fun (Last	Sigmoid
Loss function	Binary cross-entropy
Epoch	10
Batch size	32
Dense layer (neuron)	256

4.4 Experimental Scenarios

The process of building a deep learning model requires repetitive experimentations because the optimal value of the parameter and hyperparameter have a great impact on the performance. To select the best-performing model of CNN, five scenarios were used for experimentation. The scenarios used for experimentation are generalized as follows.

Scenario 1: Setting the dense layer (neuron) to 256

Table 4-2: Experimental scenario #1

Optimization algorithm	Adam	**Scenario #1**
Learning rate	0.01	
Activation fun (Last layer)	Sigmoid	
Loss function	Binary cross-entropy	
Epoch	10	
Batch size	32	
Dataset splitting	80%-20%	
Dense layer (neuron)	256	
Output		
Execution time	563.301 sec	
Training accuracy	0.9761	
Validation accuracy	0.9773	
Testing accuracy	0.9772	

Scenario 2: Configuring activation function (Softmax), number of epochs (20), and Dense layer neuron (64)

Table 4-3: Experimental scenario #2

Optimization algorithm	Adam	**Scenario #2**
Learning rate	0.01	
Activation fun (Last layer)	Softmax	
Loss function	Binary cross-entropy	
Epoch	20	
Batch size	32	
Dataset splitting	80%-20%	
Dense layer (neuron)	64	
Output		
Execution time	68.124 sec	
Training accuracy	0.260	
Validation accuracy	0.301	
Testing accuracy	0.301	

Scenario 3: Setting optimization algorithm (RMSprop), Activation Function (Sigmoid), Epoch (10).

Table 4-4: Experimental scenario #3

Optimization algorithm	RMSprop	
Learning rate	0.01	
Activation fun (Last layer)	Sigmoid	
Loss function	Binary cross-entropy	
Epoch	10	
Batch size	32	
Dataset splitting	80%-20%	
Dense layer (neuron)	64	
Output		**Scenario #3**
Execution time	605.537 sec	
Training accuracy	0.970	
Validation accuracy	0.958	
Testing accuracy	0.958	

Scenario 4: Setting data splitting 90%-10% (training and testing)

Table 4-5: Experimental scenario #4

Optimization algorithm	Adam	
Learning rate	0.01	
Activation fun (Last layer)	Sigmoid	
Loss function	Binary cross-entropy	
Epoch	10	
Batch size	32	
Dataset splitting	90%-10%	**Scenario #4**
Dense layer (neuron)	64	
Output		
Execution time	623.470 sec	
Training accuracy	0.977	
Validation accuracy	0.978	
Testing accuracy	0.978	

Scenario 5: Setting the data splitting (90%-10%) with an epoch size of 20.

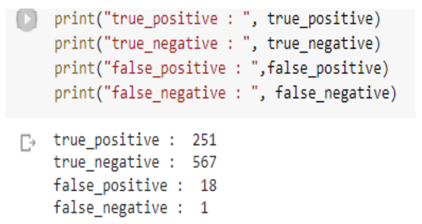
Table 4-6: Experimental scenario #5

Optimization algorithm	Adam	
Learning rate	0.01	
Activation fun (Last layer)	Sigmoid	
Loss function	Binary cross-entropy	
Epoch	20	
Batch size	32	
Dataset splitting	90%-10%	**Scenario #5**
Dense layer (neuron)	64	
Output		
Execution time	1153.151sec	
Training accuracy	0.978	
Validation accuracy	0.978	
Testing accuracy	0.978	

4.5 Experiment Results Discussion

All of the tests were conducted using the Google Colab cloud editor with the GPU accelerator turned on. Because it offers free GPU and RAM, both of which are critical for deep learning research. As mentioned in the hyperparameter selection sub-section, the study experimented on five scenarios with different parameters, hyperparameter values, and cases. Although some of the scenarios have nearly equal training, validation, and testing accuracy, the execution time varies. The scenarios with the shortest execution times were chosen.

The five scenarios assisted to select the best performing SQLI model. Among five cases, the first scenario (**Scenario #1**) achieved a score of 0.98% testing accuracy with an execution time of 563.301 sec which is a promising result. Then, the model was evaluated using the evaluation metrics mentioned in the earlier section. The optimal hyperparameter and the evaluation metrics of the proposed model are described as follows.



```
print("true_positive : ", true_positive)
print("true_negative : ", true_negative)
print("false_positive : ",false_positive)
print("false_negative : ", false_negative)
```

```
true_positive :  251
true_negative :  567
false_positive :  18
false_negative :  1
```

Figure 4-3: Scenario #1 Evaluation value

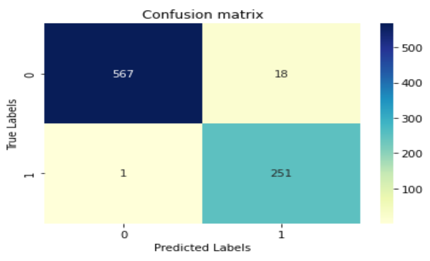
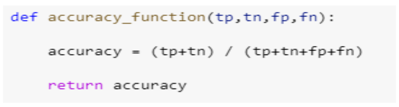


Figure 4-4: Confusion metrics for scenario #1



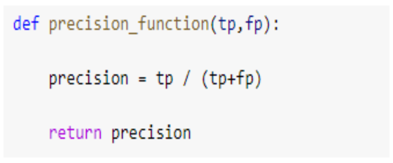
```
def accuracy_function(tp,tn,fp,fn):

    accuracy = (tp+tn) / (tp+tn+fp+fn)

    return accuracy
```

Figure 4-5: Accuracy calculation code snipet

Accuracy : 0.977



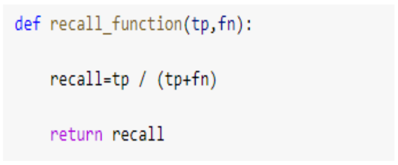
```
def precision_function(tp,fp):

    precision = tp / (tp+fp)

    return precision
```

Figure 4-6: Precision calculation code snippet

Precision : 0.933



```
def recall_function(tp,fn):

    recall=tp / (tp+fn)

    return recall
```

Figure 4-7: Recall calculation code snippet

Recall: 0.996

Table 4-7: The proposed model result summary

Proposed Model Result Summary	
Execution time	563.301 sec
Training accuracy	0.9761
Validation accuracy	0.9773
Testing accuracy	0.9772

Experiment Scenarios Summery

The process of building a deep learning model requires repetitive experiments because the optimal value of the parameter and hyperparameter have a great impact on the performance. To select the best performing model of CNN given scenarios were used for experimentation.

Table 4-8: Scenario summary

Scenarios	**Scenarios 1**	**Scenarios 2**	**Scenarios 3**	**Scenarios 4**	**Scenarios 5**
Learning rate	0.1	0.01	0.01	0.01	0.01
Activation Function	Sigmoid	Softmax	Sigmoid	Sigmoid	Sigmoid
Loss function	Binary Cross-entropy	Binary Cross-entropy	Binary Cross-entropy	Binary Cross-entropy	Binary Cross-entropy
Epoch	10	20	10	10	20
Batch Size	32	32	32	32	32
Dataset Split	80 % - 20 %	80 % - 20 %	80 % - 20 %	90 % - 10 %	90 % - 10 %
Dense Layer	256	64	64	64	64
Output Execution Time	563.301 Sec	68.1245 Sec	605.537 Sec	623.470 Sec	1153.151 Sec
Training Accuracy	0.9761 %	0.260 %	0.970 %	0.977 %	0.978%
Validation Accuracy	0.9773 %	0.301 %	0.958 %	0.978 %	0.978 %
Testing Accuracy	0.9772 %	0.301 %	0.958 %	0.978 %	0.978

Scenario 1

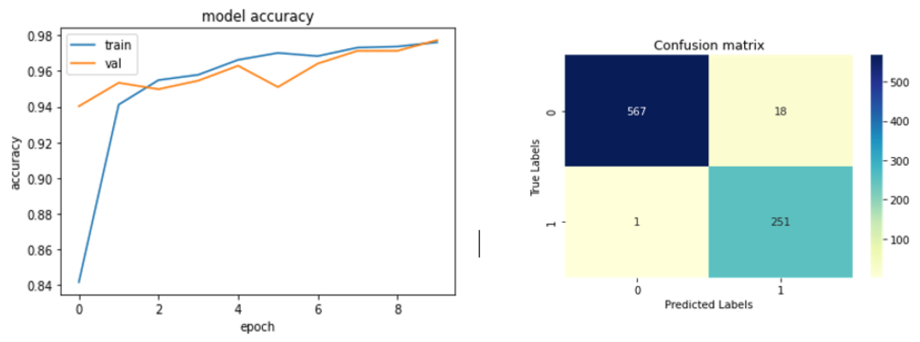


Figure 4-8: Scenario 1 visualization

Scenario 2

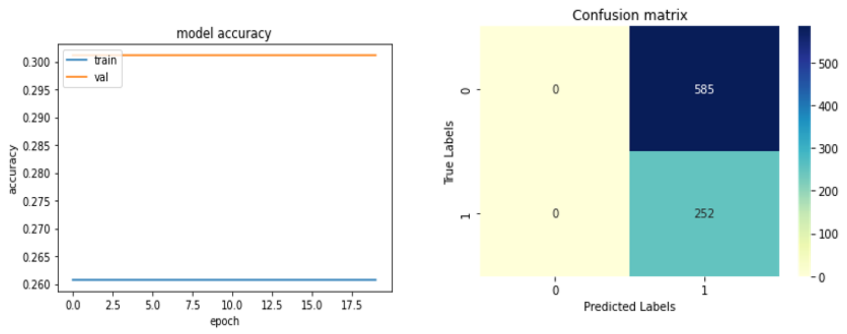


Figure 4-9: Scenario 2 visualization

Scenario 3

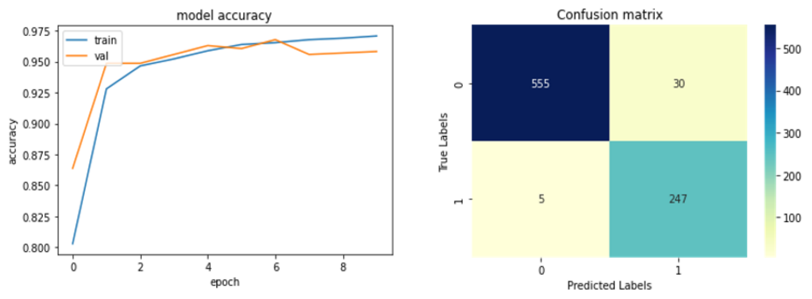


Figure 4-10: Scenario 3 visualization

Scenario 4

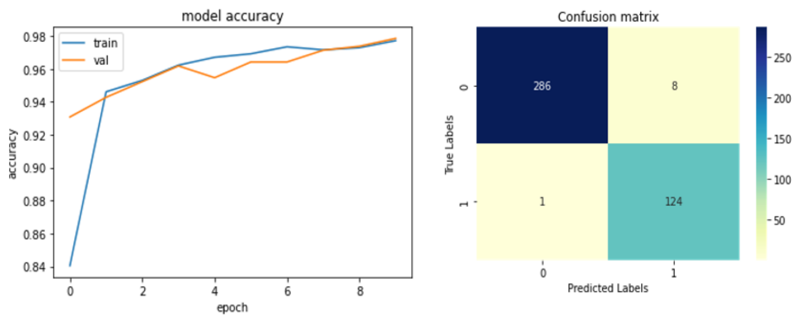


Figure 4-11: Scenario 4 visulization

Scenario 5

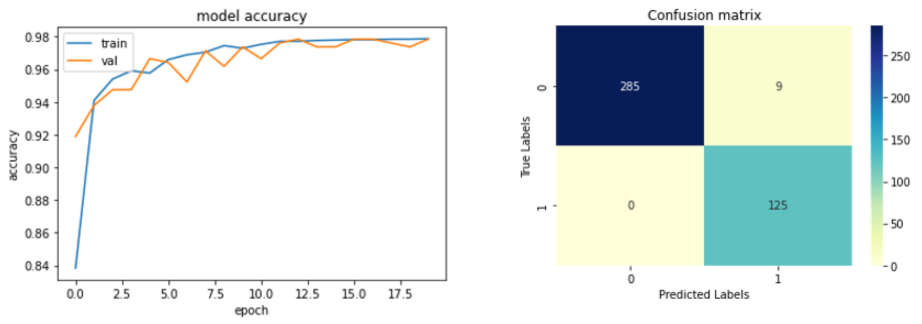


Figure 4-12: Scenario 5 visulization

One of the ways to find out whether the ML or DL models are overfitted, underfitted, or fitted is to generate the training and validation graph. The graph provides information in a way that the training and validation graph has a huge graph the model most likely overfitted, and it required some technique to solve the problem. On the other hand, if the gap between the training and validation graph is small the model is most likely fitted right. Depending on the above definition, the proposed model is fitted right. The training and validation accuracy of the proposed model is depicted in below Figure.

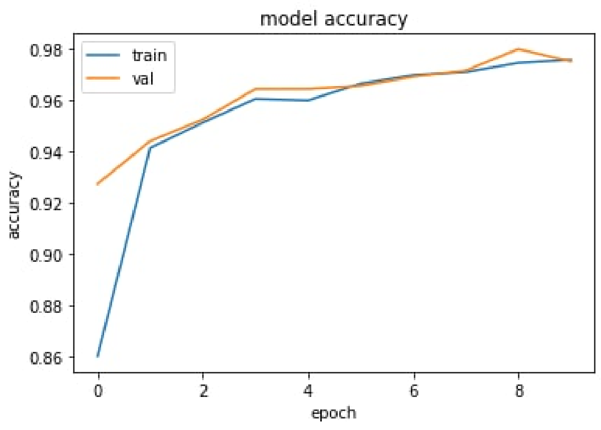
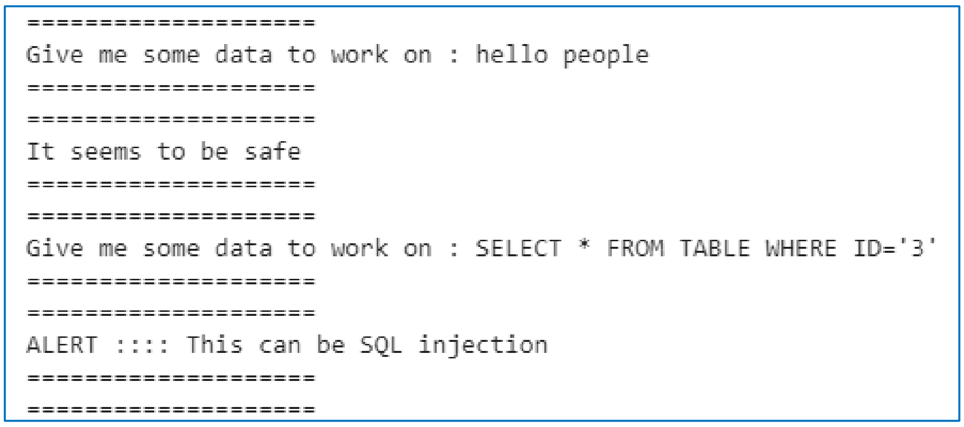


Figure 4-13: The proposed model

Here presented a sample test of the proposed SQLI prevention model.



```
====================
Give me some data to work on : hello people
====================
====================
It seems to be safe
====================
====================
Give me some data to work on : SELECT * FROM TABLE WHERE ID='3'
====================
====================
ALERT :::: This can be SQL injection
====================
====================
```

Figure 4-14: Sample testing of the proposed model

CHAPTER FIVE

CONCLUSIONS & RECOMMENDATIONS

5.1 Introduction

SQL injection attack is one of the major security challenges. As stated in the previous chapters, the SQLI attack creates a portal through which an attacker can gain direct access to a database server, allowing them to retrieve sensitive information. The attack can have varying effects depending on the database application, as well as various other possible effects. Hence, the proposed deep learning model is capable of preventing SQLIA prevention with high accuracy.

5.2 Conclusion

In this study, the CNN model was built to prevent an SQLI, and the experimentation used a public benchmark dataset. The proposed model can be used as a tool for SQL injection prevention, and also having a sufficient dataset the performance of the model will improve. The experimentation used a CNN algorithm trained with deep learning, with different hyperparameter values and scenarios. Finally, the proposed model has achieved a truly outstanding result compared to all the other scenarios. The model has achieved an execution time of **563.301 sec**, training accuracy of **0.9761**, validation accuracy of **0.9773**, and testing accuracy of **0.9772**.

5.3 Contributions

There are currently large web applications that make use of a database. As a result, SQLI has become the most common cyber security threat. In this study, a SQL injection prevention model was developed that is capable of accurately detecting and classifying cyber security threats. The proposed work focuses on preventing attacks through the use of a deep learning approach, and the results are promising. Lastly, the study's contributions are:

i. **Developing a model**: Several experimental scenarios were used to test our SQLI prevention model. The experimentation used the state-of-the-art deep learning algorithm called CNN (Convolutional Neural Network).

ii. **Preparing dataset**: Even though the dataset used for the model is a publicly available benchmark dataset, the data preprocessing and labeling was implemented on the dataset.

5.4 Recommendations

As mentioned in the above sections, the major web application which uses a database is exposed to an SQLI attack. The attack can be prevented by building a model using deep learning or machine learning. The challenging part of the experimentation was to get a sufficient dataset to train the neural network. The performance of the model depends on the amount of dataset because the more data the neural network trains the better to detect and classify the attack.

SQLI attack prevention can be achieved using different approaches and algorithms. Therefore, other researchers and students suggested exploring other deep learning algorithms like Ensemble learning, RCNN, and LSTM for the prevention of SQLI. In addition to this, hardware tools such as GPU, high-performance RAM, and CPU greatly help minimize the effort to train the neural network and produce a good result and A better results can also be derived if we combined CNN with LSTM for enhancing with the highest prediction accuracy.

References

[1] W. PETROS and K. ELHAM, "Cyber Security in the Quantum Era," *COMMUNICATIONS OF THE ACM,* vol. 62, no. 4, April 2019.

[2] D. Craigen, N. Diakun-Thibault and a. R. Purse, "Defining Cybersecurity," *Technology Innovation Management Review,* October 2014.

[3] "SQL: The Complete Reference," *James R. Groff and Paul N. Weinberg,* 1999.

[4] "https://www.fortinet.com/resources/cyberglossary/sql-injection".

[5] A. Tajpour, S. Ibrahim and M. Masrom, "SQL Injection Detection and Prevention Techniques," *University Technology Malaysia,.*

[6] Ali, Salih and Nabeel, "Investigation framework of web applications vulnerabilities, attacks and protection techniques in structured query language injection attacks," *Int. J. Wireless and Mobile Computing,,* vol. Vol. 14, no. No. 2, 2018.

[7] A. K.Kolhe and P. Adhikari, "A SQL Injection : Internal Investigation of Injection, Detection and Prevention of SQL Injection Attacks," *International Journal of Engineering Research & Technology (IJERT),* vol. 3, no. 1, January - 2014.

[8] M. Alenezi, M. Nadeem and R. Asif, "SQL Injection Attacks Countermeasures Assessments," *Indonesian Journal of Electrical Engineering and Computer Science,* vol. 21, no. 2, p. pp. 1121~1131, February 2021.

[9] Cahyadi and Nanang, "SQL injection Detection Using Deep Learning A Project Report".

[10] M. Hasan and Z. Balbahaith, "Detection of SQL Injection Attacks: A Machine Learning Approach," *International Conference on Electrical and Computing Technologies and Applications (ICECTA).*

[11] Q. LI, W. LI, J. WANG and A. M. CHENG, "A SQL Injection Detection Method Based on Adaptive Deep Forest," *SPECIAL SECTION ON DEEP LEARNING: SECURITY AND FORENSICS RESEARCH ADVANCES AND CHALLENGES,* October 17, 2019.

[12] Soomlek, K. Kamtuo and Chitsutha, "Machine Learning for SQL Injection Prevention on Server-Side Scripting," 2016.

[13] *https://www.kaggle.com/syedsaqlainhussain/sql-injection-dataset,* March 24, 2022.

[14] *https://www.delta-net.com/compliance/cyber-security/faqs/how-does-cyber-security-work,* March 24, 2022.

[15] P.S.Seemma, S.Nandhini and M.Sowmiya, "International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)," *Overview of Cyber Security,* vol. 7, no. 11, November 2018.

[16] *https://www.stackhawk.com/blog/sql-injection-prevention-spring/,* April 3, 2022.

[17] *https://www.vskills.in/certification/tutorial/sql-injection-3/,* April 6, 2022.

[18] *https://www.acunetix.com/websitesecurity/sql-injection2/,* April 12, 2022.

[19] M. Shachi, N. S. S. A. S. S. Ahmed, A. A. Brishty and N. Sakib, "A Survey on Detection and Prevention of SQL and NoSQL Injection Attack on Server-side Applications," *International Journal of Computer Applications,* vol.

183, no. 10, June 2021.

[20] N. Bhateja, D. S. Sikka and D. A. Malhotra, "A Review of SQL Injection Attack and Various Detection Approaches," *Amity University Haryana, Gurgaon, India.*

[21] M. A. Rubaiei, T. A. Yarubi, M. A. Saadi and B. Kumar, "SQLIA Detection and Prevention Techniques," *9th International Conference on System Modeling & Advancement in Research Trends,* 4th–5th, December, 2020.

[22] T. Pattewar, H. Patil, H. Patil, N. Patil, M. Taneja and T. Wadile, "Detection of SQL Injection using Machine Learning: A Survey," *International Research Journal of Engineering and Technology (IRJET),* vol. 06, no. 11, Nov 2019.

[23] C. Pinga, W. Jinshuang, Y. Lanjuan and P. Lin, "SQL Injection Teaching Based on SQLi-labs," *IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE),* 2020.

[24] W. G. Halfond, J. Viegas and A. Orso, "A Classification of SQL Injection Attacks and Countermeasures," *College of Computing Georgia Institute of Technology.*

[25] M. H. U. Sharif, "Web Attacks Analysis and Mitigation Techniques," *International Journal of Engineering Research & Technology (IJERT),* pp. 10 - 12, 2022.

[26] G. R. Chowdary, S. Neeraj and S. Sparsha, "MACHINE LEARNING APPROACHES FOR DIFFERENT CYBERTHREATS," *B.M.S Institute of technology and Management,* September 2021.

[27] I. Tafa and E. Resulaj, "h International Conference on Multidisciplinary Studies," 10-11 December 2021.

[28] Z. C. S. S. Hlaing and M. Khaing, "A Detection and Prevention Technique on SQL Injection Attacks," *Faculty of Information Science, University of Computer Studies (Magway).*

[29] Farooq and Umar, "Ensemble Machine Learning Approaches for Detection of SQL Injection Attack," *ISSN 1846-6168 (Print), ISSN 1848-5588 (Online).*

[30] M. A. Kausar, M. Nasar and A. M. Said, "SQL Injection Detection and Prevention Techniques in ASP.NET Web Application," *International Journal of Recent Technology and Engineering ·,* vol. 8, no. 3, October 2019.

[31] L. Ma, Y. Gao, D. Zhao and C. Zhao, "Research on SQL Injection Attack and Prevention Technology Based on Web," *International Conference on Computer Network, Electronic and Automation (ICCNEA),* 2019.

[32] S. Bandhakavi, P. Bisht, P. Madhusudan and V. Venkatakrishnan, "CANDID: Preventing SQL Injection Attacks using Dynamic Candidate Evaluations," *University of Illinois Chicago, Urbana-Champaign, USA.*

[33] F. Q. Kareem, S. Y. Ameen, D. M. Ahmed, S. F. Kak, I. M. I. A. M. A. Z. N. R. Hajar Maseeh Yasin and N. Omar, "SQL Injection Attacks Prevention System Technology: Review," *Asian Journal of Research in Computer Science,* vol. 10, no. 3, pp. 13-32, 2021.

[34] D. Kar, S. Panigrahi and S. Sundararajan, "SQLiDDS: SQL Injection Detection Using Query Transformation and Document Similarity," *Silicon Institute of Technology, Bhubaneswar, India,* p. 377–390, 2015.

[35] L. Xiao, S. Matsumoto, T. Ishikawa and K. Sakurai, "SQL Injection Attack Detection Method using Expectation Criterion Kyushu University," *Fourth International Symposium on Computing and Networking,* 2016.

[36] L. Ma, Y. Gao, D. Zhao and C. Zhao, "Research on SQL Injection Attack and Prevention Technology Based on Web," *International Conference on Computer Network, Electronic and Automation (ICCNEA),* 2019.

[37] R. Dorai and V. Kannan, "SQL Injection-Database Attack Revolution and Prevention," *Journal of International Commercial Law and Technology,* vol. 6, no. 4, 2011.

[38] R. Chandrashekhar, M. Mardithaya, S. Thilagam and D. Saha, "SQL Injection Attack Mechanisms and Prevention Techniques," *Springer-Verlag Berlin Heidelberg 2012,* p. 524–533, 2012.

[39] J. T. Senders, M. M. Zaki, A. V. Karhade, B. Chang, W. B. G. &. M. L. B. &. T. R. Smith and O. Arnaout, "An introduction and overview of machine learning in neurosurgical care," *REVIEW ARTICLE - NEUROSURGICAL TECHNIQUES,* 2017.

[40] Abdulmalik and Yazeed, "An Improved SQL Injection Attack Detection Model Using Machine Learning Techniques," *International Journal of Innovative Computing 11(1) 53-57*, vol. 11, no. 1, pp. 53-57, 2021.

[41] R. Mui and P. Frank, "Preventing SQL Injection through Automatic Query Sanitization with ASSIST," *Polytechnic Institute of NYU,* p. 27–38, 2010.

[42] *https://www.ptsecurity.com/ww-en/analytics/knowledge-base/how-to-prevent-sql-injection-attacks/#2,* May 23, 2022.

[43] B. S. Kumar and P. Anaswara, "Vulnerability detection and prevention of SQL injection," *International Journal of Engineering & Technology,* vol. 7, pp. 16-18, January 2018.

[44] *https://www.allroundautomations.com/products/pl-sql-developer/?gclid=CjwKCAiA1JGRBhBSEiwAxXblwYu7LEpBuotCpcGlZq_y2jCE7MzKX8UE4IwQKR_VdKzhGH-YSUwcEhoCZoMQAvD_BwE,* April 26, 2022.

[45] M. Bravenboer, E. Dolstra and E. Visser, "Preventing injection attacks with syntax embeddings," *Elsevier,* October 2007.

[46] Krüger, R. A. McClure and I. H., "SQL DOM: Compile Time Checking of Dynamic SQL Statements," May 2005.

[47] Cahyadi and Nanang, "SQL injection Detection Using Deep Learning A Project Report," *Sekolah Elektro dan Informatika Institut Teknologi Bandung.*

[48] *https://www.geeksforgeeks.org/role-based-access-control/,* April 29, 2022.

[49] *https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html,* April 29, 2022.

[50] C. Torrano-Gimenez, A. Perez-Villegas and G. Alvarez, "An Anomaly-Based Approach for Intrusion Detection in Web Traffic," *Instituto de Fisica Aplicada, Consejo Superior de Investigaciones Cientificas.*

[51] MWARUWA and M. CHAKA, "LONG SHORT TERM MEMORY BASED DETECTION OF WEB BASED SQL INJECTION ATTACKS," *UNIVERSITY OF NAIROBI,* p. 52, 2017.

[52] S. Singh and A. Kumar, "Detection and Prevention of SQL Injection," *International Journal of Scientific Research & Engineering Trends,* vol. 6, no. 3, 2020.

[53] M. Martin, B. Livshits and M. S. Lam, "Finding Application Errors and Security Flaws Using PQL: a Program Query Language," *Computer Science Departmen, Stanford University,* 2005.

[54] Rauti and Sampsa, "Towards Cyber Attribution by Deception," *University of Turku, Finland.*

[55] J. Hasan, A. M. Zeki, A. Alharam and N. Al-Mashhur, "Evaluation of SQL Injection Prevention Methods," *International Conference on Modeling Simulation and Applied Optimization (ICMSAO),* 2019.

[56] *https://auth0.com/blog/hashing-passwords-one-way-road-to-security/,* April 29, 2022.

[57] *https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html,* April 30, 2022.

[58] S. Gadgil, S. Pillai and S. Poojary, "SQL INJECTION ATTACKS AND PREVENTION TECHNIQUES," *International Journal on Recent and Innovation Trends in Computing and Communication,* vol. 1, no. 4, pp. 293-296, April 2013.

[59] Robinson, M. Akbar and M. A. F. Ridha, "SQL Injection and Cross Site Scripting Prevention Using OWASP Web Application Firewall," *INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION,* vol. 2, no. 4, 2018.

[60] Murphy, I. E. Naqa and M. J., "What Is Machine Learning?," *Machine Learning in Radiation Oncology: Theory and Applications.*

[61] J. Alzubi, A. Nayyar and A. Kumar, "Machine Learning from Theory to Algorithms: An Overview," *Second National Conference on Computational Intelligence (NCCI 2018),* 2018.

[62] A. SHRESTHA and A. MAHMOOD, "Review of Deep Learning Algorithms and Architectures," *IEEE,* April 1, 2019.

[63] Nasteski and Vladimir, "An overview of the supervised machine learning methods," *Faculty of Information and Communication Technologies, Partizanska bb,* December 2018..

[64] *https://softwareengineering.stackexchange.com/questions/368671/training-data-in-unsupervised-learning,* April 30, 2022.

[65] J. T. Senders, M. M. Zaki, A. V. Karhade, B. Chang, W. B. Gormley, M. L. Broekman, T. R. Smith and O. A. , "An introduction and overview of machine learning in neurosurgical care," *REVIEW ARTICLE - NEUROSURGICAL TECHNIQUES,* pp. 29-38, 2018.

[66] Mahesh and Batta, "Machine Learning Algorithms - A Review," *International Journal of Science and Research (IJSR),* 2018.

[67] Chapelle, O. Schölkopf and A. B. Zien, "Book Reviews," *IEEE TRANSACTIONS ON NEURAL NETWORKS,* vol. 20, no. 3, MARCH 2009.

[68] G. Huang, S. Song, J. N. D. Gupta and C. Wu, "Semi-Supervised and Unsupervised Extreme Learning Machines," *IEEE TRANSACTIONS ON CYBERNETICS,* 2015.

[69] Sarker and I. H., "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *REVIEW ARTICLE,* 29 May 2021.

[70] Q. LI, W. LI, J. WANG and M. CHENG, "A SQL Injection Detection Method Based on Adaptive Deep Forest," *SPECIAL SECTION ON DEEP LEARNING: SECURITY AND FORENSICS RESEARCH ADVANCES AND CHALLENGES,* October 17, 2019.

[71] A. SHRESTHA and A. MAHMOOD, "Review of Deep Learning Algorithms and Architectures," *IEEE,* April 1, 2019.

[72] Sastry, Upadhya, Vidyadhar and P. S., "An Overview of Restricted Boltzmann Machines," *Rreview Article,* vol. 99, no. 2, June 2019.

[73] COŞKUN, Musab, YILDIRIM, Özal, UÇAR, Ayşegül, DEMIR and Yakup, "AN OVERVIEW OF POPULAR

DEEP LEARNING METHODS," *European Journal of Technic EJT,* vol. 7, no. 2, 2017.

[74] Indolia, Sakshi, Goswami, A. Kumar, Mishra, S. P., Asopa and Pooja, "Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach," *International Conference on Computational Intelligence and Data Science (ICCIDS 2018),* pp. 679-688, 2018.

[75] *https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/,* April 30, 2022.

[76] *https://www.tutorialspoint.com/python_deep_learning/python_deep_learning_deep_neural_networks.htm,* April 30, 2022.

[77] M. Jiang, Y. Liang, X. Feng, X. Fan, Z. Pei, Y. Xue and R. Guan, "Text classification based on deep belief network and softmax regression," *RECENT ADVANCES IN PATTERN RECOGNITION AND ARTIFICIAL INTELLIGENCE,* 2016.

[78] H. Zhang, T. Huang, S. Liu, H. Yin, J. Li, H. Yang and Y. Xia, "A learning style classification approach based on deep belief network for large-scale online education," *Journal of Cloud Computing: Advances, Systems and Applications,* vol. 9, no. 26, 2020.

[79] D. S. Smys, D. J. I. Z. Chen and D. S. Shakya, "Survey on Neural Network Architectures with Deep Learning," *Journal of Soft Computing Paradigm (JSCP) (2020),* vol. 2, no. 3, pp. 186-194, 2020.

[80] Williams, S. Thomas and Laurie, "Using automated fix generation to secure SQL statements," *29th International Conference on Software Engineering Workshops(ICSEW'07),* 2007.

[81] Z. Lashkaripour and A. G. Bafghi, "A Security Analysis Tool For Web Application Reinforcement Against SQL Injection Attacks (SQLIAs)," *International ISC Conference on Information SECURITY & CRYPTOLOGY, ·* August 2013.

[82] P. SAVCBS, "Specification and Verification of Component-Based Systems," *12th ACM SIGSOFT Symposium on the Foundations of Software Engineering,* October 31-November 5, 2004.

[83] Chen, X. Fu, X. Lu and B. P. Shijun, "A Static Analysis Framework For Detecting SQL Injection Vulnerabilities," *31st Annual International Computer Software and Applications Conference(COMPSAC 2007),* 2007.

[84] D. Appelt, C. D. Nguyen, L. Briand and N. Alshahwan, "Automated Testing for SQL Injection Vulnerabilities: An Input Mutation Approach," *Conference Paper,* July 2014.

[85] Orso, W. G. Halfond and Alessandro, "Preventing SQL Injection Attacks Using AMNESIA," *College of Computing Georgia Institute of Technology,* 2006.

[86] Ghafarian and D. Ahmad, "A Hybrid Method for Detection and Prevention of SQL Injection Attacks," *Computing Conference,* 2017.

[87] I. Lee, S. Jeong, S. Yeo and J. Moon, "A novel method for SQL injection attack detection based on removing SQL query attribute values," *Mathematical and Computer Modelling,* vol. 55, no. 1-2, pp. 58-68, 2012.

[88] R. Mui and P. Frankl, "Preventing SQL Injection through Automatic Query Sanitization with ASSIST," *Salaun, Fu, and Hall ¨ e (Eds.): Fourth International Workshop on Testing, Analysis and Verification of Web Software,* pp. 27-38, 2010.

[89] Shiva, R. Dharam and S. G., "Runtime Monitoring Technique to handle Tautology based SQL Injection Attacks,"

International Journal of Cyber-Security and Digital Forensics (IJCSDF) 1(3):, vol. 1, no. 3, pp. 189-203, 2012.

[90] W. Qing and C. He, "The research of an AOP-based approach to the detection and defense of SQL injection attack," *International Conference on Advanced Electronic Science and Technology (AEST 2016),* 2016.

[91] M. E. Hossain and S. Ahmed, "An approach to secure multi-tier websites through SQL-Injection detection and prevention," *ICCA*, Jan 2020.

[92] K. Ross, "SQL Injection Detection Using Machine Learning Techniques and Multiple Data Sources," *San Jose State University SJSU ScholarWorks,* 2018.

[93] A. Alam, M. Tahreen, M. M. Alam, S. A. Mohammad and S. Rana, "SCAMM: Detection and Prevention of SQL Injection Attacks," *Department of Computer Science and Engineering BRAC University,* 2021.

[94] S. Mishra, "SQL Injection Detection Using Machine Learning," *The Faculty of the Department of Computer Science San José State University,* 2019.

[95] I. Jemal, H. Hamam, O. Cheikhrouhou and A. Mahfoudhi, "SQL Injection Attack Detection and Prevention Techniques Using Machine Learning," *International Journal of Applied Engineering Research*, January 2020.

[96] T.P.Latchoumi, M. S. Reddy and K.Balamurugan, "Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention," *European Journal of Molecular & Clinical Medicine,* vol. 07, no. 02, 2020.

[97] N. Cahyadi, "SQL injection Detection Using Deep Learning A Project Report," *Rekayasa dan Manajemen Keamanan Informasi Sekolah Elektro dan Informatika Institut Teknologi Bandung.*

[98] S. ABAIMOV and G. BIANCHI, "CODDLE: Code-Injection Detection With Deep Learning," *University of Rome Tor Vergata, Rome, Italy,* 2019.

[99] D. Chen, Q. Yan, C. Wu and J. Zhao, "SQL Injection Attack Detection and Prevention Techniques Using Deep Learning," *Journal of Physics: Conference Series,* 2020.

[100] K. Zhang, "A Machine Learning based Approach to Identify SQL Injection Vulnerabilities," *IEEE/ACM International Conference on Automated Software Engineering (ASE),* 2019.

[101] M. M. Hassan, R. B. Ahmad and T. Ghosh, "SQL Injection Vulnerability Detection Using Deep Learning: A Feature-based Approach," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI),* vol. 9, no. 3, p. 702~718, September 2021.

[102] A. Luo, W. Huang and W. Fan, "A CNN-based Approach to the Detection of SQL Injection Attacks," *IEEE,* 2019.

[103] M. Hirani, A. Falor, H. Vedant, P. Mehta and D. Krishnan, "A Deep Learning Approach for Detection of SQL Injection Attacks using Convolutional Neural Networks," *Department of Computer Engineering,MPSTME, NMIMS University, Mumbai, India,* 2020.

[104] J. K. R, P. Beriwal, S. B. B, A. Amarajan and N. Pandey, "An Efficient SQL Injection Detection System Using Deep Learning," *International Conference on Computational Intelligence and Knowledge Economy (ICCIKE),* 2021.

[105] T. GHOSH, "SQL Injection Vulnerability Detection using Deep Learning: A Web Feature Based Approach,"

Department of Software Engineering DAFFODIL INTERNATIONAL UNIVERSITY, 2019.

[106] J. Wang, Z. Zhou and J. Chen, "Evaluating CNN and LSTM for Web Attack Detection," *Association for Computing Machinery.,* 2018.

[107] H. Zhang, J. Zhao, B. Zhao, X. Yan, H. Yuan and F. Li, "SQL Injection Detection Based on Deep Belief Network," *CSAE2019.*

[108] M. Zhang, B. Xu, S. Bai, S. Lu and Z. Lin, "A Deep Learning Method to Detect Web Attacks Using a Specially Designed CNN," *Springer International Publishing,* p. 828–836, 2017.

[109] Y. Fang, J. Peng, C. Huang and L. Liu, "WOVSQLI: Detection of SQL Injection Behaviors Using Word Vector and LSTM," *Association for Computing Machinery.,* 2018.

[110] *https://scikit-learn.org/stable/modules/feature_selection.html,* April 30, 2022.

[111] *https://in.mathworks.com/discovery/feature-extraction.html,* May 1, 2022.

[112] *https://www.infoworld.com/article/3003315/deep-learning-a-brief-guide-for-practical-problem-solvers.html,* May 1, 2022.

[113] R. Yamashita, M. Nishio, R. K. G. Do and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging,* vol. 9, p. 611–629, 2018.

[114] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," *arXiv,* 2 Dec 2015.

[115] H. Gholamalinezhad and H. Khosravi, "Pooling Methods in Deep Neural Networks, a Review," *Ph.D. Student of Electronics - Image Processing, Faculty of Electrical & Robotics Engineering, Shahrood University of Technology, Daneshgah Blvd., Shahrood, Iran..*

[116] A. Ajit, K. Acharya and A. Samanta, "A Review of Convolutional Neural Networks," *International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE),* February 2020.

[117] V. T. Ponnada and S. N. Srinivasu, "Efficient CNN for Lung Cancer Detection," *International Journal of Recent Technology and Engineering (IJRTE),* vol. 8, no. 2, July 2019.

[118] P. A. Abhay K.Kolhe, "A SQL Injection : Internal Investigation of Injection, Detection and Prevention of SQL Injection Attacks," *International Journal of Engineering Research & Technology (IJERT),* vol. 3, no. 1, January - 2014.

[119] I. Jemal, O. Cheikhrouhou, H. Hamam and A. Mahfoudhi, "SQL Injection Attack Detection and Prevention Techniques Using Machine Learning," *International Journal of Applied Engineering Research ·,* vol. 15, no. 6, pp. 569-580, January 2020.

[120] I. Jemal, O. Cheikhrouhou, H. H. and A. Mahfoudhi, "SQL Injection Attack Detection and Prevention Techniques Using Machine Learning," *International Journal of Applied Engineering Research,* vol. 15, no. 6, pp. 569-580, 2020.

Appendix

```
#importing python library
import glob
import time
import pandas as pd
from nltk import ngrams
from nltk.tokenize import sent_tokenize
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.stem import PorterStemmer
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
```

```
from google.colab import drive
drive.mount('/content/drive')


#importing a keras deep learning framework
from tensorflow import keras


#importing other ML libraries
import pandas as pd
import os


#importing numpy libraries
import numpy as np



#reading csv dataset from mounted google drive
df=pd.read_csv('/content/drive/MyDrive/SQLI/sqli_modified.csv',encoding='utf-
8')
```

```
#Importing train test splitting library
from sklearn.model_selection import train_test_split
#Splitting the dataset to 80%-20%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rand
om_state=42)



# Importing all necessary libraries to build a CNN neural network
import tensorflow as tf
from keras.models import Sequential
from keras import layers
from keras.preprocessing.text import Tokenizer
from keras.wrappers.scikit_learn import KerasClassifier


## Preparing the CNN architecture
model=tf.keras.models.Sequential([

    tf.keras.layers.Conv2D(64, (3,3), activation=tf.nn.relu, input_shape=(64,
64,1)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation=tf.nn.relu),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(256, (3,3), activation=tf.nn.relu),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256,activation='relu'),
    ##tf.keras.layers.Dense(128,activation='relu'),
    ##tf.keras.layers.Dense(64,activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])


#Compiling model
model.compile(loss='binary_crossentropy',
              optimizer='adam',



              metrics=['accuracy'])
model.summary()
```