

CS850- DATABASE SECURITY

A REPORT ON THE PROJECT ENTITLED

“SECURING WEB APPLICATION DATABASES FROM SQL INJECTION ATTACKS”



SUBMITTED BY:

Anik kanti Biswas

232IS005

II SEMESTER M-TECH CSE-IS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL

2023 – 2024

Abstract

SQL (structure question language) injection is one among the hazards to database-connected programmes, like Web-based apps, mobile applications, and even desktop applications. By victimization SQL injection, AN wrongdoer will acquire complete access to AN application or info, permitting them to get rid of or edit sensitive knowledge while not permission. SQL injection could be a vulnerability in applications that don't adequately validate the user's input. SQL Injection Attacks (SQLIA) occur once AN wrongdoer manipulates user computer file to enter a series of malicious SQL statements into a question for execution by the back-end info. Applications can be promptly hacked victimization this kind of threat, permitting the wrongdoer to get confidential knowledge. This study compares and contrasts classical and fashionable design.

Keywords: ,Keywords— SQL injection, Database security, SQLMAP, Nikto, Web application ,DVWA, CNN.

1 Introduction

The Internet is currently a wide used info infrastructure. The increasing quantity of information saved in databases has recently been impressed by the multiplied use and fast advancement of the net infrastructure. The importance of spatially securing information or info concerning business, company businesses, establish- ments, organisations, varied money transactions, health info, personal info, and alternative internet-based services has multiplied because the variety of users and their reliance on digital info has big. All on-line apps will be accessed via the net mistreatment any browser on any software package. net applications became the foremost extensively used interface for retrieving or inserting these information or info. SQL injection attacks pose a significant security risk to Web applications because they provide attackers with unrestricted access to the databases that enable the apps, as well as the extremely sensitive data contained in those databases. Despite the fact that various solutions to handle the SQL injection problem have been given by researchers and practitioners, existing approaches either fail to address the complete scope of the problem or have limitations that prevent their adoption and acceptance. We'll look at PHP tactics and other methods for preventing SQL injection, as well as methods for detecting SQL attacks, types of SQL injection, causes of SQL injection via receiving and publishing, and SQL vulnerability protection technologies in this article. SQL injection flaws in web applications can provide an attacker complete access to the application's databases. Because these databases typically contain sensitive client or user information, security violations such as identity theft, private information loss, and fraud can occur. In rare cases, attackers can utilise a SQL injection vulnerability to gain control of and harm the system that serves the Web application.

1.1 Problem Description

SQL injections result in the theft of sensitive data. It also has the potential to be disastrous for any company, government, or organisation. Such incidents can damage a company's operations and image, as well as result in significant fines imposed by data protection laws. In the next section, we will explore some critical prevention and detection methodology in order to prevent such a highly damaging attack.

1.2 Motivation

SQL injection is a type of insertion attack that is defined as a damaging strategy that involves injecting malicious SQL statements or exploiting defective input into a website's SQL-based application software. It does not necessitate high-level implementation, resulting in one of the most lethal assaults ever. SQL injection attacks are always at the top of the priority list for attackers. SQL has steadily made its way into a number of industrial and open-source databases. SQL injection (SQLi) is a type of cyber security attack that targets these databases by using specially prepared SQL statements to fool the systems into performing unexpected and unintended actions. It usually enables the assailant to look at information that they would not otherwise be able to access. SQL has steadily found its way into various industry and open supply databases since its inception. SQL injection (SQLi) is a type of cybersecurity attack that exploits specially constructed SQL statements to deceive these databases into doing unexpected and unintended activities.

1.3 Scope

SQL Injection is a technique for convincing an application to run SQL statements that were not expected. The attacker's purpose is to get beyond the security screening by exploiting the results of the modified SQL query to gain unlawful access. The majority of online apps are vulnerable to this form of attack. With careful planning and execution, such dangers can be avoided. Now that we're aware of the problems, we're putting in place the necessary validation checks to prevent SQL injection. When we are able to prevent such attacks, most web applications will be less vulnerable to attack, at least in terms of their databases. There may be more strategies for preventing such attacks, so we may delve deeper into this to identify the best-suited and most powerful security wall against all injection attack.

1.4 Objectives

SQL injection is a type of internet security flaw that allows an attacker to meddle with the queries that a programme conducts on its data. It usually allows an offender to look at information that they wouldn't otherwise be able to access. SQL injection, often known as SQLI, is a typical attack vector that employs malicious SQL code to manipulate backend data and get access to information that was not intended to be revealed. This information could include everything from sensitive company information to user lists to personal client information. The purpose of this idea is self-evident. To begin, we identified a common cause and vulnerability in the system, such as inadequate user input validation. To solve this problem, we proposed a set of coding principles that encourage defensive coding techniques such encoding and validation of user input. Preventing SQL injection concerns with a comprehensive and systematic application of these solutions is a great way to go. However, in fact, such tactics are carried out by humans, making them prone to errors. Repairing outdated codebases that may contain SQL injection vulnerabilities can also be time-consuming and labor-intensive. Using a CNN model to detect various types of SQL injection, we can deploy the model into the website before any input is processed. This proactive approach allows us to identify and prevent potential SQL injection attacks by analyzing input data for malicious patterns or behaviors. By integrating the CNN model into the system's input validation process, we can significantly reduce the risk of SQL injection vulnerabilities and enhance the overall security posture of the application.

1.5 Research Gap

Deep learning in SQL injection detection uses CNNs for image-based analysis, RNNs for sequences, and hybrid models for spatial-temporal patterns. Techniques include transfer learning, unsupervised learning for anomaly detection, and adversarial training for robustness against evasion, enhancing cybersecurity in web applications.

2 Related Work

To gain adequate information concerning SQL injection attacks, I reviewed variety of electronic journal articles from IEEE journals and ACM, additionally as gathered some data from websites. The papers listed below cover numerous vital methods for preventing SQL injection attacks.

1. I listed the techniques for sql injection discovery in "Using analyse Tree Validation to stop SQL Injection Attacks" ACM. The SQL analyse tree validation that I discussed in my report (Gregory T. Buehrer, Bruce W. Weide, and Paolo A. G. Sivilotti, 2005) was conjointly well listed during this paper.
2. They listed the techniques to examine and sanitise input question mistreatment SQLCHECK, that uses increased queries and SQLCHECK descriptive linguistics to validate question, in "The Essence of Command Injection Attacks in net

Applications" ACM. (2006, Zhendong Su and port of entry Wassermann).

3. They coated a quick background, SQL statements, and vulnerability replacement ways from IEEE CNF's "Using automatic Fix Generation to Secure SQL Statements" (Stephen Thomas and Laurie Williams, 2007).

4. They lined an artless methodology to mechanically shield applications from SQL injection attacks in "Automated Protection of PHP Applications against SQL-injection Attacks." To secure existing properties, the initial approach combines static analysis, dynamic analysis, and automatic code re-engineering (Et-tore Merlo, Dominic Letarte, Giuliano Antoniol, 2007).

5. They additionally provided a unique approach to shield hold on procedures from attack and sight SQL injection from sit in "Preventing SQL Injection Attacks in hold on Procedures" (Ke dynasty, M. Muthuprasanna, Suraj Kothari, 2007). This technique combines a runtime talk to a static analysis of the appli- ance code to eliminate attack vulnerabilities. This attack works by dynamic the structure of the initial SQL statement and sleuthing the SQL injection attack. the tactic is split into 2 phases, one offline and therefore the different runtime. hold on procedures use a programme within the offline section to pre-process and sight SQL statements within the execution entail runtime analysis. The tech- nique controlled all runtime generated SQL queries associated with user input throughout the runtime section and checked these against the initial structure of the SQL statement once receiving input from the user. once this system detects malicious SQL statements, it prevents them from accessing the info and provides data regarding the attack.

3 The Proposed Methodology

An SQL Injection attack can be carried out on any website's login page to get access to the user's account or to retrieve all of the database's data. There are a variety of scripts that can be used to implement SQL Injection. When an at- tacker knows the username but not the password and wants to get access to a specific user, there are primarily two scenarios. In this example, the attacker types the username and replaces the password with a script to get access to the user's account and do whatever actions are desired. The second scenario is when an attacker is unfamiliar with a single user's credentials but is ready to query the database for all user information. The attacker also added a similar script to the username and password columns in this example. Code injection occurs when a malicious script is injected into a web application from an external source, such as an input field provided by the web application to receive user input. This attack takes advantage of the fact that there isn't any accurate data validation for input and output. The harmful code was embedded inside the software and operates as part of it. A successful code injection attack could cause asset dam- age or unwanted behaviour.

Different Types of SQL Injection Attack which can be performed:

1. SQL injection attack for a specific user

In this type of attack, the attacker will try to gain access to a specific user's account.

Username: `validU serName'OR'1 = 1andPassword : (NoInput)Ex :
vikas'OR'1 = 1`

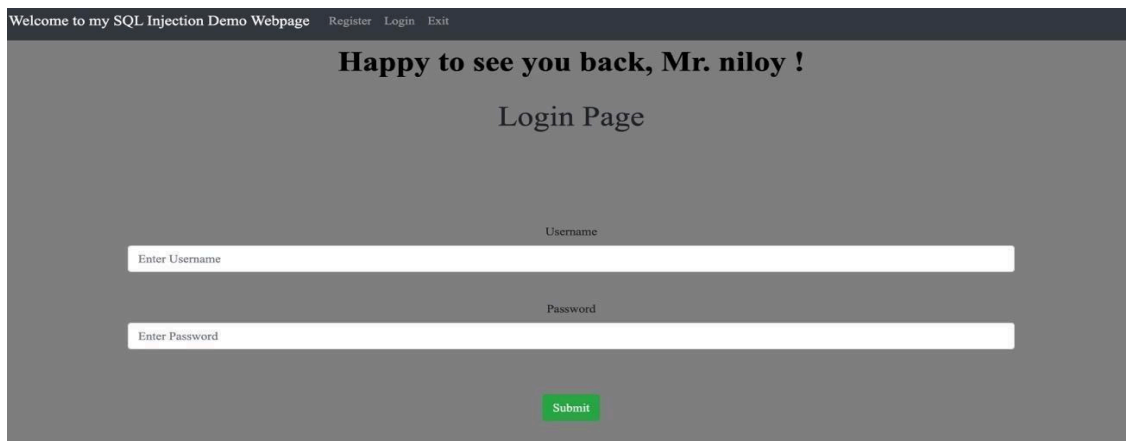


Fig. 1. SQL Injection attack

The password script in the above query is set up so that the password attribute produces a Boolean result based on a Boolean expression like `pass- word=or 1=1`, which is always true since `1=1` yields a true value. An attacker can successfully log into the user's account using this approach.

1. SQL Injection attack to get all data form the database

In this type of attack, the attacker will try to gain the access of all user's ac- counts. Username: `' or '1'='1` Password: `' or '1'='1`

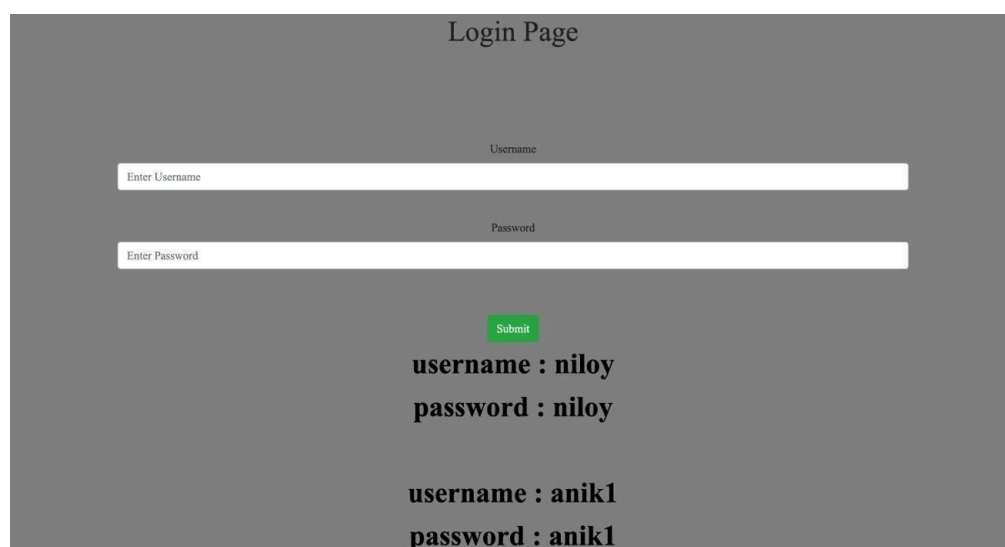


Fig. 2. SQL Injection attack

Both the username and password scripts in the above query are built so that they are equal and yield a Boolean result based on Boolean expressions like `username = 1 or 1-1 AND password=or 1=1` which is always true because `1-1` returns a true value. The WHERE clause returns true for every row, allowing the attacker to get all database records and gain access to the application. An attacker can successfully retrieve all data from the user's database using this method. Now the next step will be creating a DL model classify SQLi based input.

4 Result and discussion

Nikto Scanning:

```
kali@kali: ~  
File Actions Edit View Help  
~$ nikto -host https://akanik0.wordpress.com  
- Nikto v2.5.0  
  
+ Multiple IPs found: 192.0.78.13, 192.0.78.12  
+ Target IP: 192.0.78.13  
+ Target Hostname: akanik0.wordpress.com  
+ Target Port: 443  
  
+ SSL Info: Subject: /CN=*.wordpress.com  
Ciphers: TLS_AES_256_GCM_SHA384  
Issuer: /C=GB/ST=Greater Manchester/L=Salford/O= Sectigo Limited/CN=Sectigo ECC Domain Validation Secure Server CA  
+ Start Time: 2024-02-25 13:09:41 (GMT-5)  
  
+ Server: nginx  
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options  
+ /: Drupal Link header found with value: <https://wp.me/Pfy2W4-h>; rel=shortlink. See: https://www.drupal.org/  
+ /: Uncommon header 'x-hacker' found, with contents: Want root? Visit join.a8c.com/hacker and mention this header.  
+ /: Uncommon header 'host-header' found, with contents: WordPress.com.  
+ /: Uncommon header 'x-ac' found, with contents: 1.bom_dca HIT.  
+ /: An alt-svc header was found which is advertising HTTP/3. The endpoint is: ':443'. Nikto cannot test HTTP/3 over QUIC. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/alt-svc  
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
```

SQLMAP Penetration Testing :

```
Shell No. 1  
File Actions Edit View Help  
$ sqlmap --wizard  
  
[!]: legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
[*] starting @ 13:17:25 /2024-02-25/  
  
[13:17:25] [INFO] starting wizard interface  
Please enter full target URL (-u): https://akanik0.wordpress.com/  
POST data (--data) [Enter for None]:  
[13:17:43] [WARNING] no GET and/or POST parameter(s) found for testing (e.g. GET parameter 'id' in 'http://www.site.com/vuln.php?id=1'). Will search for forms  
Injection difficulty (--level/--risk). Please choose:  
[1] Normal (default)  
[2] Medium  
[3] Hard  
> 1  
Enumeration (--banner/--current-user/etc). Please choose:  
[1] Basic (default)  
[2] Intermediate  
[3] All  
> 1
```

5 Work done till now

Some pre-requisite software and language must be familiarised with/installed on the system in order to accomplish the SQL Injection Demo Attack. The programme described above, as well as the instructions to begin the activity, are listed below.

First we have to install the XAMPP Server.

- 1.1. Now we have to Open XAMPP Control Panel.
- 1.2. Now we have to Start Apache and MySQL.

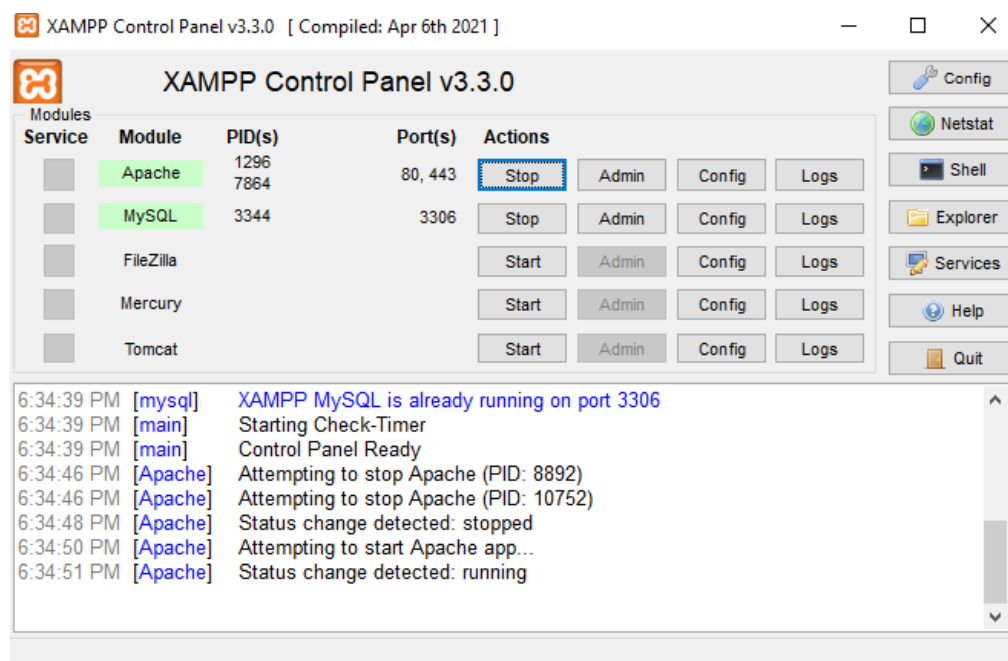
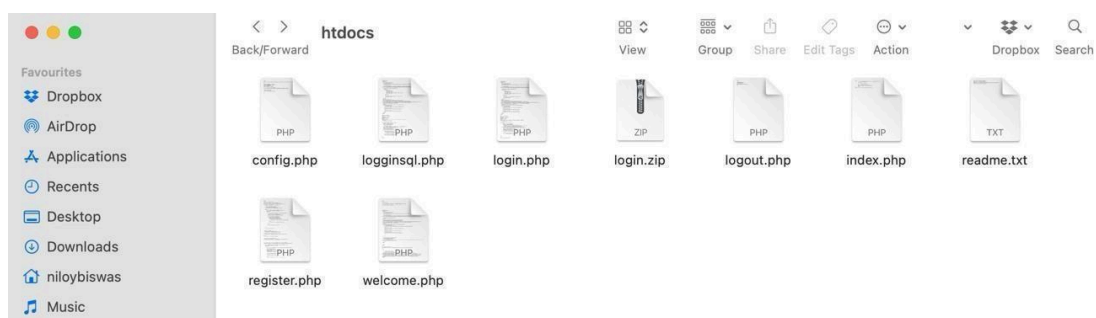


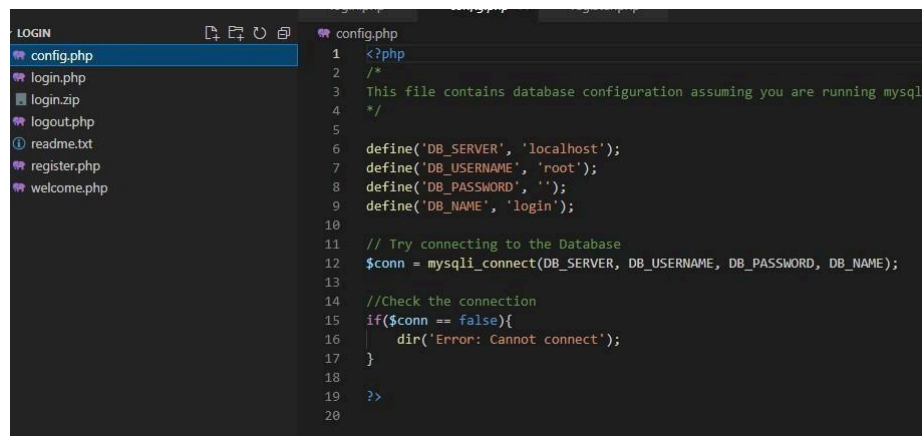
Fig. 3. XAMPP Control Panel

Create different PHP files for login, logout, config, and register.

- 1.3. Create a PHP file in the location: C:

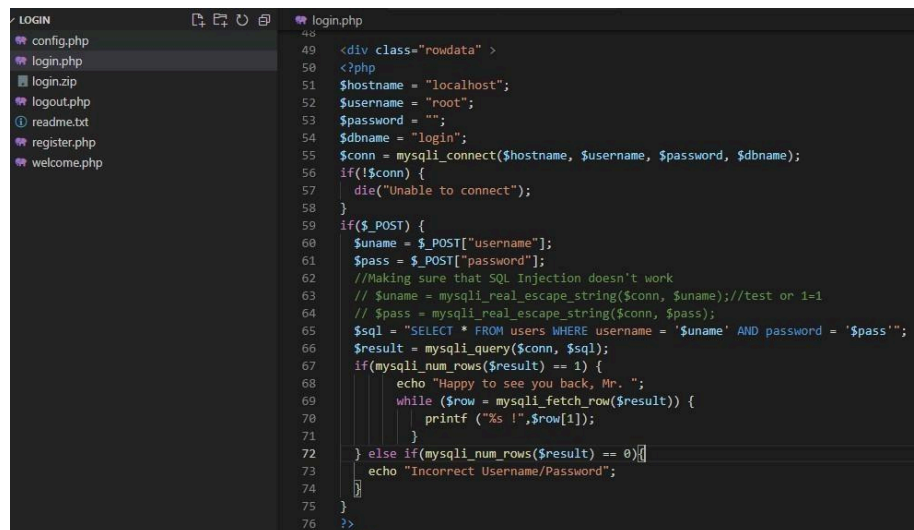


Open the folder in VS Code Editor.



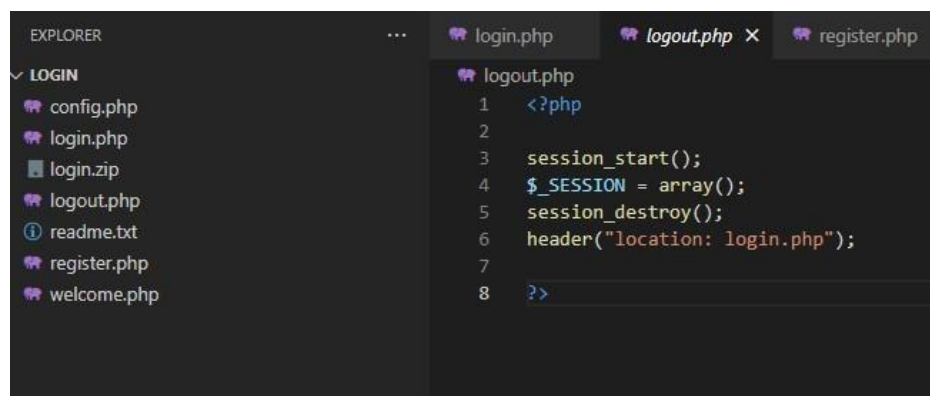
```
1 <?php
2 /*
3  This file contains database configuration assuming you are running mysql
4  */
5
6  define('DB_SERVER', 'localhost');
7  define('DB_USERNAME', 'root');
8  define('DB_PASSWORD', '');
9  define('DB_NAME', 'login');
10
11  // Try connecting to the Database
12  $conn = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_NAME);
13
14  //Check the connection
15  if($conn == false){
16      die('Error: Cannot connect');
17  }
18
19  ?>
```

Fig. 5. config.php



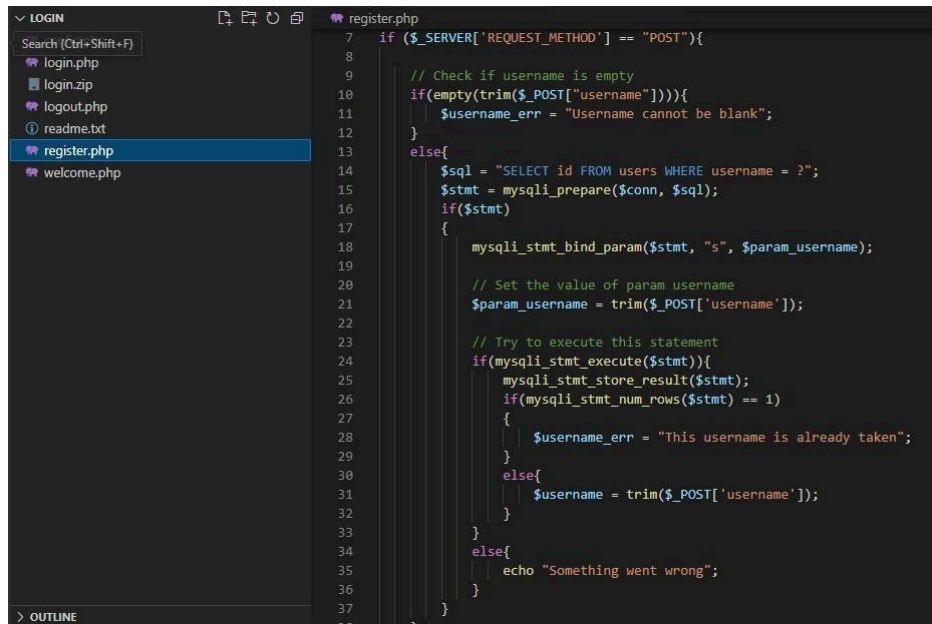
```
48
49 <div class="rowdata" >
50 <?php
51 $hostname = "localhost";
52 $username = "root";
53 $password = "";
54 $dbname = "login";
55 $conn = mysqli_connect($hostname, $username, $password, $dbname);
56 if(!$conn) {
57     die("Unable to connect");
58 }
59 if($_POST) {
60     $uname = $_POST["username"];
61     $pass = $_POST["password"];
62     //Making sure that SQL Injection doesn't work
63     // $uname = mysqli_real_escape_string($conn, $uname);//test or 1=1
64     // $pass = mysqli_real_escape_string($conn, $pass);
65     $sql = "SELECT * FROM users WHERE username = '$uname' AND password = '$pass'";
66     $result = mysqli_query($conn, $sql);
67     if(mysqli_num_rows($result) == 1) {
68         echo "Happy to see you back, Mr. ";
69         while ($row = mysqli_fetch_row($result)) {
70             printf ("%s !", $row[1]);
71         }
72     } else if(mysqli_num_rows($result) == 0){
73         echo "Incorrect Username/Password";
74     }
75 }
76 ?>
```

Fig. 6. login.php



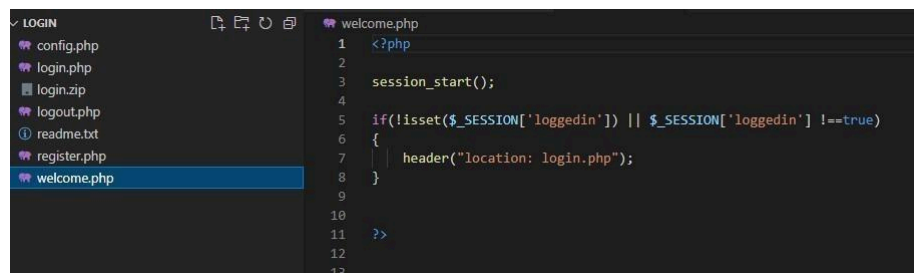
```
1 <?php
2
3 session_start();
4 $_SESSION = array();
5 session_destroy();
6 header("location: login.php");
7
8 ?>
```

Fig. 7. logout.php



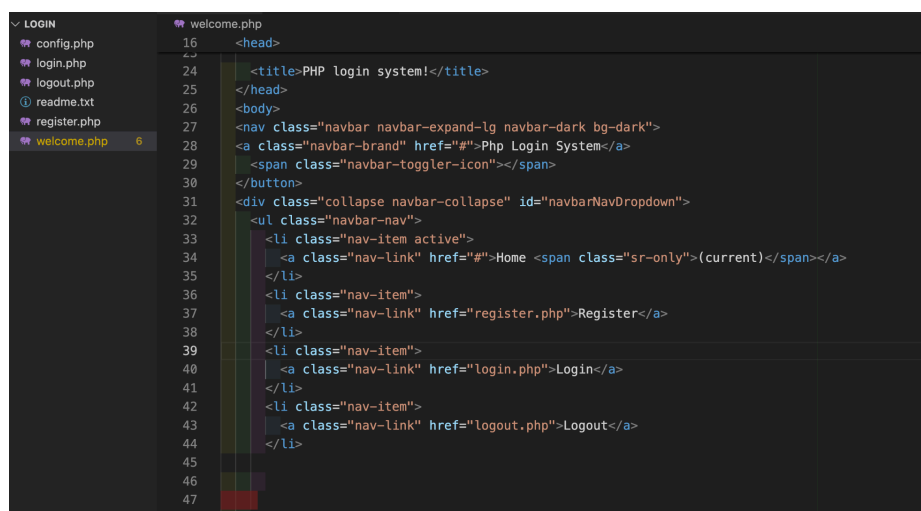
```
7 if ($_SERVER['REQUEST_METHOD'] == "POST"){
8
9     // Check if username is empty
10    if(empty(trim($_POST["username"]))){
11        $username_err = "Username cannot be blank";
12    }
13    else{
14        $sql = "SELECT id FROM users WHERE username = ?";
15        $stmt = mysqli_prepare($conn, $sql);
16        if($stmt)
17        {
18            mysqli_stmt_bind_param($stmt, "s", $param_username);
19
20            // Set the value of param username
21            $param_username = trim($_POST['username']);
22
23            // Try to execute this statement
24            if(mysqli_stmt_execute($stmt)){
25                mysqli_stmt_store_result($stmt);
26                if(mysqli_stmt_num_rows($stmt) == 1)
27                {
28                    $username_err = "This username is already taken";
29                }
30                else{
31                    $username = trim($_POST['username']);
32                }
33            }
34            else{
35                echo "Something went wrong";
36            }
37        }
38    }
```

Fig. 8. register.php



```
1 <?php
2
3 session_start();
4
5 if(!isset($_SESSION['loggedin']) || $_SESSION['loggedin'] !==true)
6 {
7     header("location: login.php");
8 }
9
10
11 ?>
```

Fig. 9. welcome.php

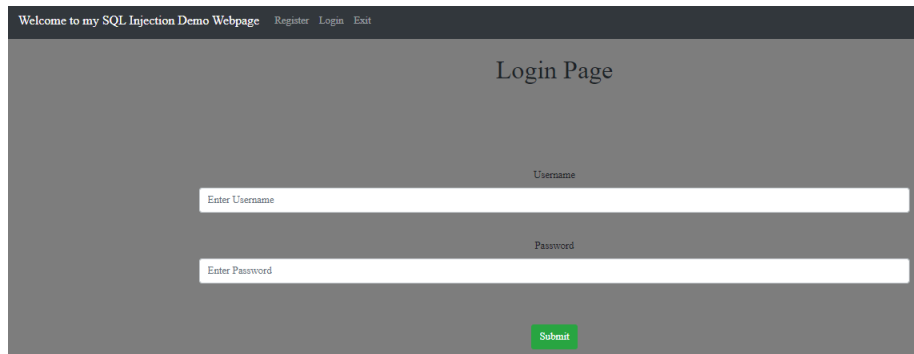


```
16 <head>
17
18 <title>PHP login system!</title>
19 </head>
20 <body>
21 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
22 <a class="navbar-brand" href="#">Php Login System</a>
23 <span class="navbar-toggler-icon"></span>
24 </button>
25 <div class="collapse navbar-collapse" id="navbarNavDropdown">
26 <ul class="navbar-nav">
27 <li class="nav-item active">
28 <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
29 </li>
30 <li class="nav-item">
31 <a class="nav-link" href="register.php">Register</a>
32 </li>
33 <li class="nav-item">
34 <a class="nav-link" href="login.php">Login</a>
35 </li>
36 <li class="nav-item">
37 <a class="nav-link" href="logout.php">Logout</a>
38 </li>
39 </ul>
40 </div>
```

Fig. 10. welcome.php

Open Localhost in the browser.

- 3.1 open : <http://localhost/login/login.php>



Welcome to my SQL Injection Demo Webpage Register Login Exit

Login Page

Username

Enter Username

Password

Enter Password

Submit

Fig. 11. login page

- 3.2 open : <http://localhost/phpmyadmin/index.php>

Create a Database and table in PHPMYAdmin.

- 1.4. Database Name: login
- 1.5. Table Name: users

Server: localhost » Database: login » Table: users

Browse

Structure

SQL

Search

Insert

Export

Import

Privileges

Operations

Tracking

Triggers

Showing rows 0 - 1 (2 total, Query took 0.0003 seconds.)

SELECT * FROM `users`

☐ Profiling

[Edit inline]

[Edit]

[Explain SQL]

[Create PHP code]

[Refresh]

☐ Show all

|

Number of rows: 25

|

Filter rows: Search this table

|

Sort by key: None

Extra options

↶

↷

id

username

password

created_at

☐

Edit

Copy

Delete

1

niloy

niloy

2024-01-21 18:47:17

☐

Edit

Copy

Delete

2

anik1

anik1

2024-01-21 18:48:46

↶

☐ Check all

With selected:

Edit

Copy

Delete

Export

☐ Show all

|

Number of rows: 25

|

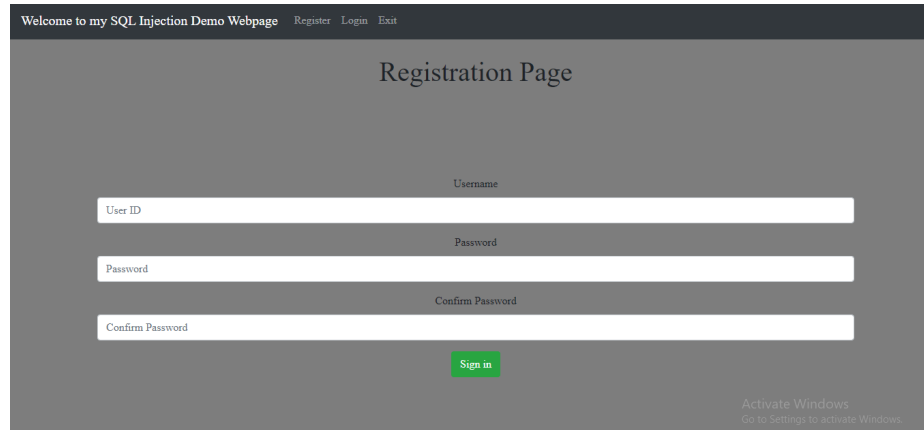
Filter rows: Search this table

|

Sort by key: None

Fig. 12. Database

5. Register a user from the registration page.



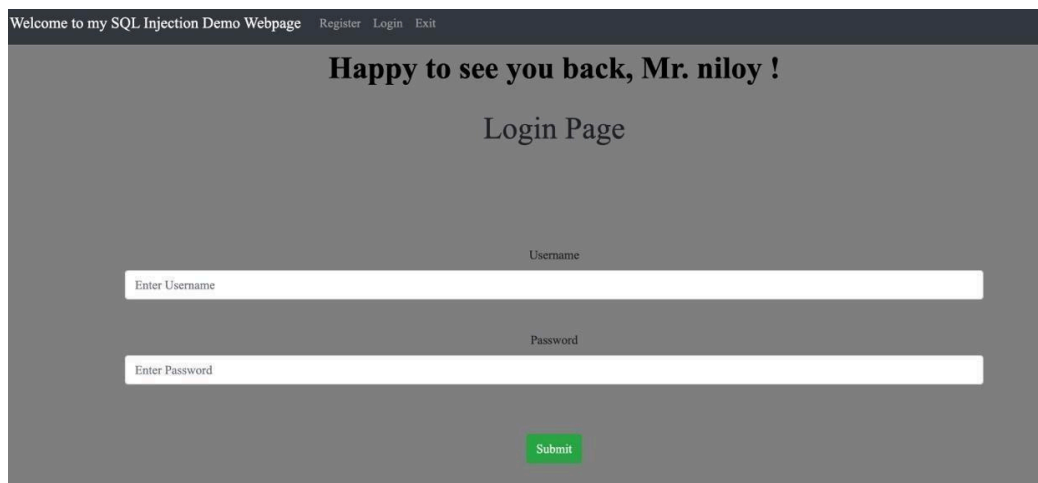
The screenshot shows a web application interface for a registration page. At the top, a dark header bar contains the text "Welcome to my SQL Injection Demo Webpage" followed by links for "Register", "Login", and "Exit". Below the header, the page title "Registration Page" is centered. The main content area has a light gray background and contains three input fields: "User ID", "Password", and "Confirm Password". Each field is a white rectangle with its label above it. Below the "Confirm Password" field is a green button labeled "Sign in". In the bottom right corner, there is a small watermark that says "Activate Windows Go to Settings to activate Windows."

Fig. 13. Registration page

6. SQL injection attack for a specific user

In this type of attack, the attacker will try to gain access to a specific user's account.

Username: valid *UserName'OR'1 = 1andPassword : (NoInput)Ex :
kaushik'OR'1 == '1';*



The screenshot shows the login page of the same web application. The header is identical to the registration page. The main content area has a light gray background and displays a message "Happy to see you back, Mr. niloy !" in bold black text. Below the message, the page title "Login Page" is centered. There are two input fields: "Enter Username" and "Enter Password". Each field is a white rectangle with its label above it. Below the "Enter Password" field is a green button labeled "Submit".

Fig. 14. SQL Injection attack

The password script in the above query is set up so that the password attribute produces a Boolean result based on a Boolean expression like `pass- word=or 1=1`, which is always true since `1=1` yields a true value. An attacker can successfully log into the user's account using this approach.

SQL Injection attack to get all data from the database

In this type of attack, the attacker will try to gain access of all user's accounts.

Username: ' or '1'='1 Password: ' or '1'='1

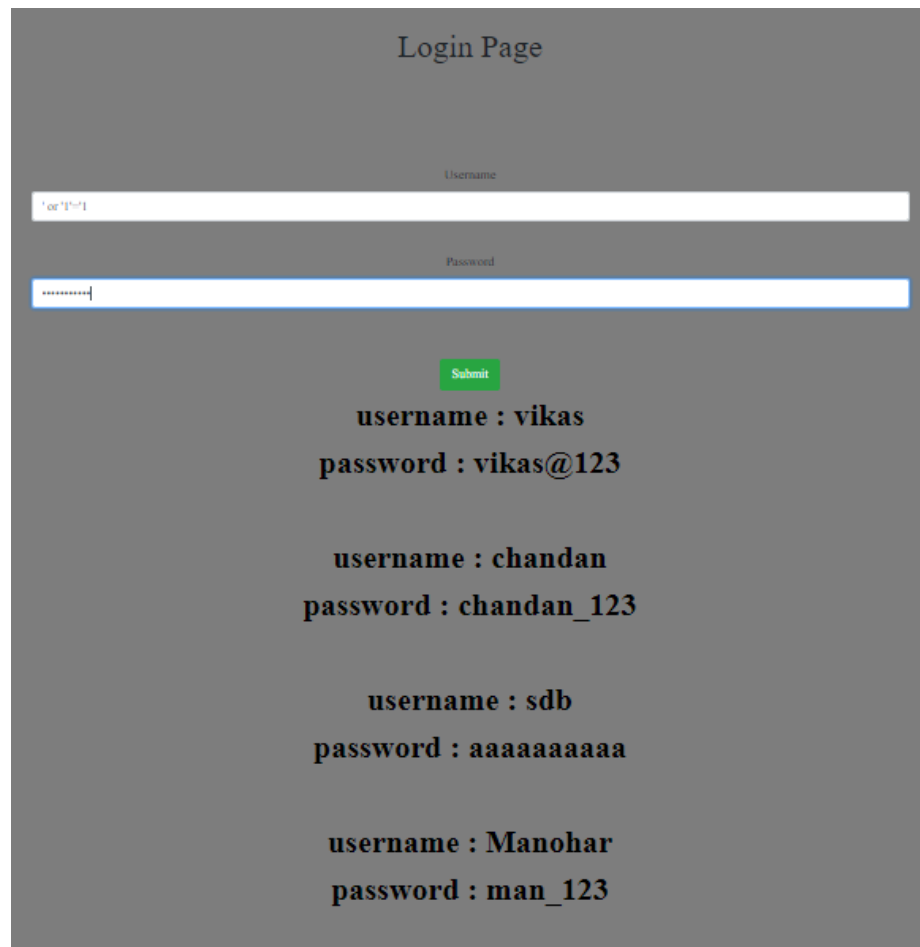


Fig. 15. SQL Injection attack

METHOD

Now, we propose a SQL Injection attacks detection method based on convolutional neural network. In this report, the traffic received by the victim host is used as the research object. Through the data collection and sanitization of the traffic, the payload of the SQL injection attacks is carved. After constructing the CNN network model, payload data can be used as the input data for model detection to report if the traffic contains SQL Injection attacks.

A. Data Collection

The purpose of this project is to detect the SQL injection behavior information contained in the traffic. Therefore, in the step of data collection, we use the packets transmitted by the network received by the victim host as the primitive data, which contains complete traffic information.

B. Data Sanitization

The traffic data contains complete traffic information that needs to be cleaned to remove information that is not relevant to the study. The specific process includes two steps. Firstly, we should extract event information from the complete traffic data, and obtaining corresponding log data, and then we need extract the corresponding URL payload from the log data to complete the data cleaning work. After the data cleaning is completed, we finally extract the valid information needed for the detection.

C. Data Normalization

Through the previous data acquisition and sanitization operations, we finally get the payload data in string format. In order to convert it into the legal input data format for the CNN model, a further vectorization processing is needed.

1) URLdecode Processing

After getting the payload raw data, we need to perform URLdecode processing to convert the data in string format into an array of word strings.

2) Training Vectorization Model

After URLdecode processing, we use the mature toolkit gensim to perform word to vector training. The payload data processed by URLdecode processing constitutes the corpus as training data, so as to obtain the trained word vector model.

3) Vectorization

Using the trained word vector model, each string element in the array is vectorized to obtain a float array with a normalized value range. Finally, the string array obtained by URLdecode processing will be converted into an array consisted of float numbers. The resulting array is used as an input data for the convolutional neural network model.

D. Building CNN Model

A convolutional neural network is a kind of feedforward neural network. Its artificial neurons can respond to a part of the surrounding elements in the coverage, extract the features of the data through convolution functions and extract highdimensional features from the low-dimensional features of the data. It has excellent performance in large image processing. After the flow data is vectorized, the flow data can be converted into the format of the image data, which can be easily processed by CNN.

CNN networks are highly similar to ordinary neural networks, and they all consist of neurons with learnable weights and bias constants. Each neuron receives some input and does some dot product calculations, and the output is the score for each category. Some calculation techniques in the common neural network are still applicable here. The convolutional layer is a unique structure of the CNN network. Each convolutional layer is composed of several convolutional units, and the parameters of each convolutional unit are optimized by a back propagation algorithm. The purpose of the convolution operation is to extract different features of the input. The lower layer convolution layer may only extract some low-level features, and more layers of the network can iteratively extract more complex features from the lower-level features.

To make the CNN model work, we should define the format of the input data. And the format is defined by multiplying the maximum feature-length obtained during the data vectorization process by the maximum feature number.

As shown in Figure 16, the complete CNN model should include several convolutional and pooling layers, a fully connected layer, or an additional hidden layer to refine the model. It will finally get the output by processing the input data.

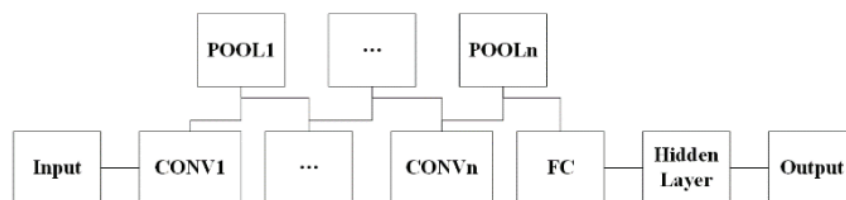


Fig. 16. CNN General Model

- Convolutional layer : The role of this layer is for convolution calculation of data.
- Pooling layer : This layer is used for feature dimensionality reduction, which can compress the number of data and parameters, reduce over-fitting, and improve the fault tolerance of the model.
- Fully connected layer : The function of this layer is to connect all the features and send the output value to the next layer.
- Hidden layer : The role of this layer is to make the final processing before passing the output value to the classifier, further reducing the possibility of data overfitting, thus improving the generalization ability of the classifier.

EXPERIMENT

A. Dataset

In this experiment, we collected SQLIAs malicious traffic and normal traffic from several aspects. The data used in the experiment is mainly divided into two parts, one is the training dataset used to train the CNN model, and the other is the real data used to compare the detection efficiency of the CNN model and the traditional method ModSecurity.

To avoid the common over-fitting issue in the field of machine learning, we build training datasets and test datasets from totally mutual exclusive different data sources. What's more, we use both manual conducted SQL injection techniques and automatic hacking tools to generate real attack data for building the validation dataset. In order to avoid the attack type being too single in the attack process, we also randomly extract small amounts of data from the training set for the improvement and supplement of the test set.

TABLE I. COMPOSITION OF EXPERIMENT DATASET

	SQL Injection Payloads	Normal Payloads
Training Data	4187	785
Test Data	1016	232

As shown in Table, the training dataset includes 4187 SQL injection payloads and 785 normal payloads. The verification data set includes 1016 SQL injection payloads and 232 normal payloads.

B. Data Processing

In this experiment, we used the DVWA experimental environment and SQLmap for real attacks and achieving the expected results of a successful data dump of the database. At the same time, we implemented a crawler script for DVWA web page emulated normal access. The tcp dump tool on the victim host is used to capture and save the network traffic, which completes the data collection operation of the experiment.

C. Building CNN Model

Sequential Model: The `tf.keras.models.Sequential` function is used to create a sequential neural network model. In this type of model, layers are added sequentially, one after the other.

Convolutional Layers:

First Convolutional Layer (Conv2D): This layer has 64 filters, each with a size of 3x3. The activation function used is ReLU (Rectified Linear Unit). The input shape for this layer is (64,64,1), indicating that the input images are expected to be grayscale images of size 64x64 pixels.

MaxPooling Layer (MaxPooling2D): This layer performs max pooling with a pool size of 2x2, reducing the spatial dimensions of the feature maps by half.

Second Convolutional Layer (Conv2D) and MaxPooling Layer (MaxPooling2D):

Similar to the first convolutional layer, the second convolutional layer has 128 filters with a size of 3x3 and uses ReLU activation. It is followed by a max pooling layer with a pool size of 2x2.

Third Convolutional Layer (Conv2D) and MaxPooling Layer (MaxPooling2D):

The third convolutional layer has 256 filters with a size of 3x3 and uses ReLU activation. It is followed by a max pooling layer with a pool size of 2x2.

Flatten Layer: This layer flattens the 3D output from the convolutional layers into a 1D array, preparing it for input into the densely connected layers.

Densely Connected Layers (Dense):

First Dense Layer: This layer has 256 neurons and uses ReLU activation.

Second Dense Layer: It has 128 neurons with ReLU activation.

Third Dense Layer: This layer has 64 neurons with ReLU activation.

Output Layer (Dense): The output layer has 1 neuron with a sigmoid activation function, which is suitable for binary classification problems. It outputs a probability score indicating the likelihood of the input belonging to one class (e.g., positive class in binary classification).

Model Compilation (compile):

Loss Function: Binary cross-entropy is used as the loss function, suitable for binary classification tasks.

Optimizer: The Adam optimizer is used, which is an adaptive learning rate optimization algorithm.

Metrics: The model's performance will be evaluated based on accuracy during training and evaluation.

Overall, this model is designed for binary image classification tasks, such as distinguishing between two classes based on input grayscale images of size 64x64 pixels. It uses convolutional layers for feature extraction and dense layers for classification, with ReLU activation functions in most layers and binary cross-entropy as the loss function.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 64)	640
max_pooling2d (MaxPooling2D)	(None, 31, 31, 64)	0
conv2d_1 (Conv2D)	(None, 29, 29, 128)	73856
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 128)	0
conv2d_2 (Conv2D)	(None, 12, 12, 256)	295168
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 256)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 256)	2359552
dense_1 (Dense)	(None, 128)	32896
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 1)	65

=====
Total params: 2,770,433
Trainable params: 2,770,433
Non-trainable params: 0
=====

```
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
```

```
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='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=tf.nn.relu),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

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

Training The Model

The code snippet demonstrated below is training of the CNN model, using the fit method with the specified parameters. The X train and y train datasets are used for training, while X test and y test are used for validation during training. The model is trained for 30 epochs, where each epoch represents one complete pass through the training dataset. The verbose = True parameter indicates that training progress will be displayed during each epoch. Additionally, a batch size of 128 is used, meaning that the training data is divided into batches of 128 samples, and the model's weights are updated after processing each batch. Throughout training, the model's performance metrics, such as accuracy, loss, and validation metrics, are monitored to assess its learning progress and generalization ability on unseen data.

```
classifier_nn = model.fit(X_train,y_train,
                          epochs=30,
                          verbose=True,
                          validation_data=(X_test, y_test),
                          batch_size=128)
```

EVALUATION

Evaluation Metrics:

In this experiment, we will use the four indicators of Accuracy, Precision, Recall, and F1 as the evaluation criteria for the model detection results. The formulas for the four evaluation indicators are calculated as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1 = \frac{2TP}{2TP+FP+FN} \quad (4)$$

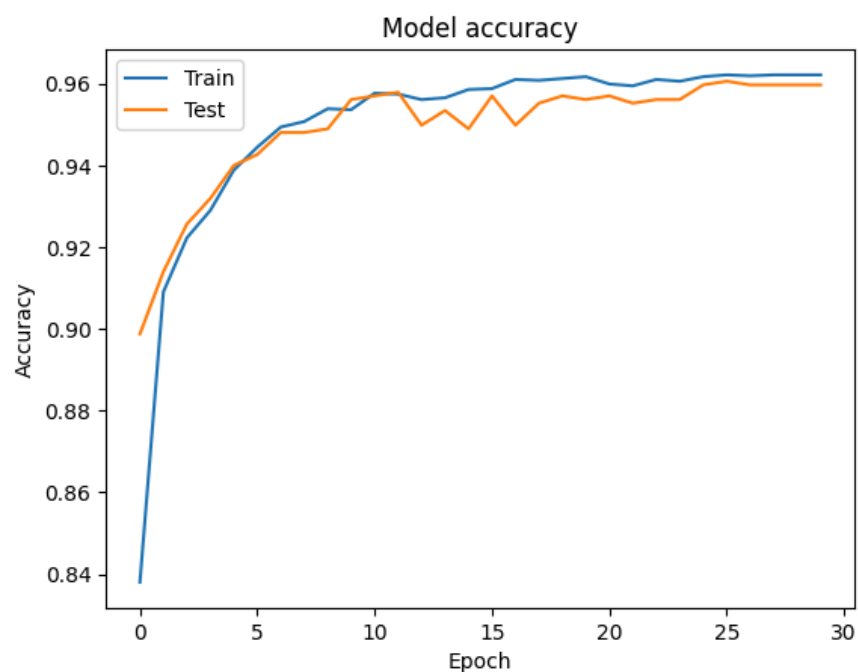
In these formulas, TP is short for True Positives, which means that malicious traffic is identified as malicious traffic, TN is short for True Negatives, which means that normal traffic is identified as normal traffic, FP is short for False Positives, which means that normal traffic is identified as malicious traffic, FN is short for False Negatives, which means that malicious traffic is identified as normal traffic.

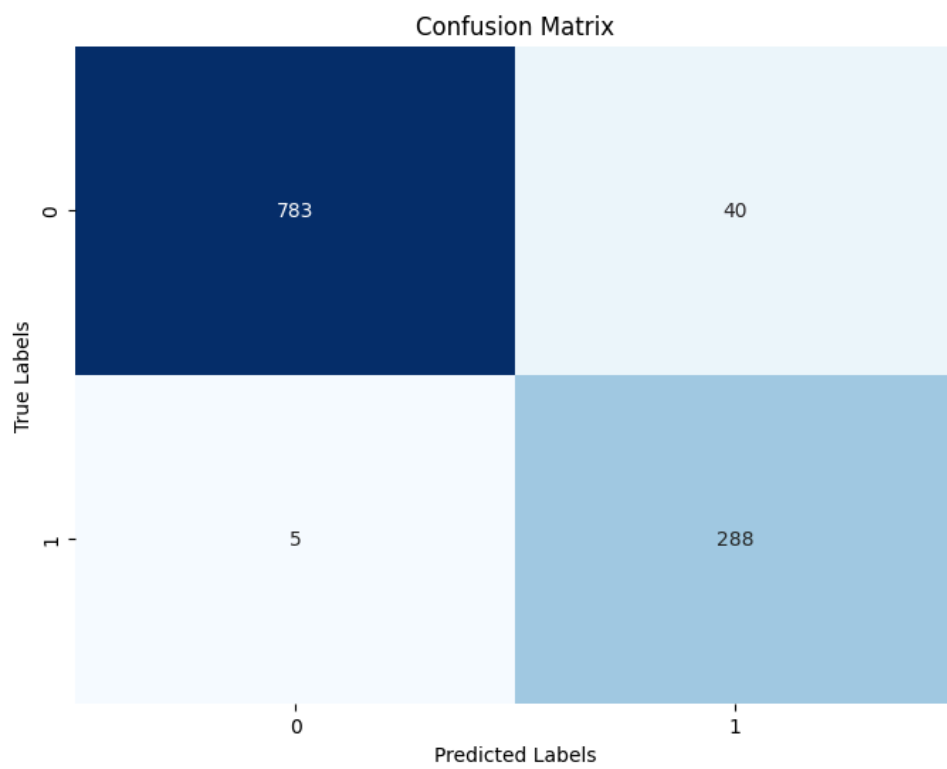
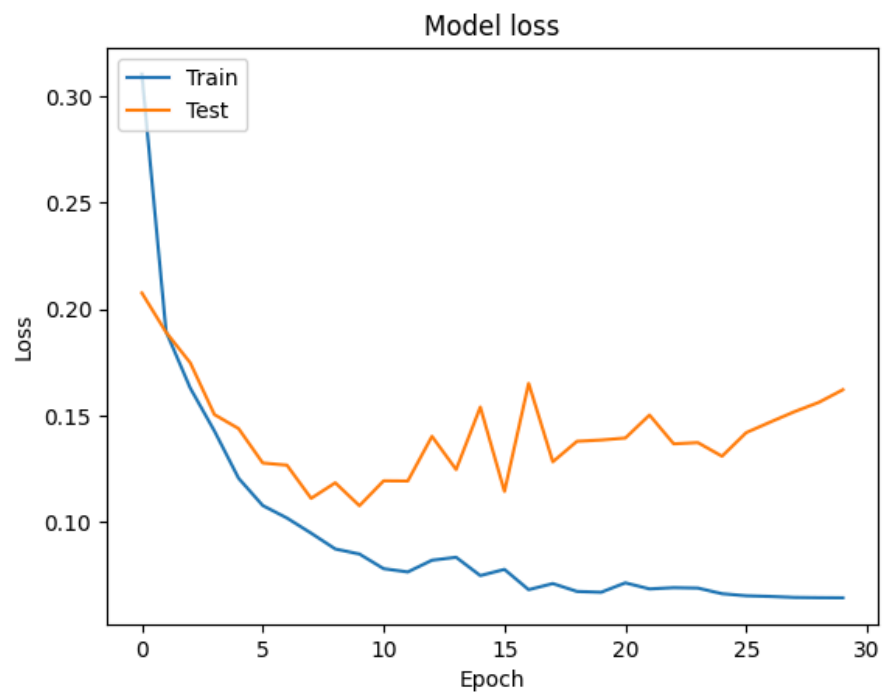
In essence, the accuracy rate represents the proportion of all samples with correct predictions to the total sample size; the precision rate indicates how many predicted positive samples are real positive; the recall rate represents how many are correctly predicted in the real positive examples in the sample; F1 is the harmonic mean of precision and recall. Through the above several indicators, we can comprehensively examine some aspects of the performance of the classifier.

Experimental Results :

TABLE II. EVALUATION RESULTS OF SQLIAS

	Accuracy	Precision	Recall	F1 Score
CNN	95.56	87.80	98.29	93.04





6 Conclusion

Code injection attacks, particularly SQL injection attacks, are one of the most well-known problems. Controlling malicious SQL code/script on an online application while maintaining end-user privacy is still a significant challenge for web developers. These issues must be addressed by web developers working on database-driven websites. By compromising a web application, an attacker can use SQL injection to get confidential information from a database. Various SQL injection defense strategies have been proposed.

It can be seen from the experimental results that among the various indicators, the detection method based on CNN model has certain improvement compared with the rule-matchingbased method. It is proved that the SQL injection detection method based on the CNN model is effective and feasible. Thus the model can be deployed in any website for input validation and detection of any malicious query.

References

1. O. Ojagbule, H. Wimmer and R. J. Haddad, "Vulnerability Analysis of Content Management Systems to SQL Injection Using SQLMAP," SoutheastCon 2018, St. Petersburg, FL, USA, 2018, pp. 1-7, doi: 10.1109/SECON.2018.8479130.
2. Phillips C., Demurjian S., Ting T.C.:Safety and Liveness for an RBAC/MAC security model. In Proceedings of the Data and Applications Security XVII, pp. 316– 329, 2004.
3. Li N., Tripunitara M.V.:Security Analysis in Role-Based Access Control. ACM Transactions on Information and System Security, vol. 9, no. 4, pp. 391—420, 2006.
4. Rakkay H., Boucheneb Security Analysis of Role Based Access Control Models Using Colored Petri Nets and CPNtools. Transactions on Computational Science IV, pp. 149–176, 2009.
5. Ferrera A.L., Madhusudan P., Parlato Security Analysis of Role-Based Access Control through Program Verification. In Proceedings of the IEEE 25th Computer Security Foundations Symposium, pp., 2012.
6. Martin E., Tau X.:Inferring Access-Control Policy Properties via Machine Learning. In Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks, pp. , 2006.
7. Singh M.P., Sural S., Atluri V., Vaidya J., Yakub U.:Managing Multi-dimensional Multi-granular Security Policies Using Data Warehousing. In Proceedings of the International Conference on Network and System Security, pp. 221–235, 2015.
8. A survey of access control methods, National Institute of Standards and Technology, and National Security Agency. Technical report, 2009.
9. Jin X., Krishnan R., Sandhu R. unified attribute-based access control model covering DAC, MAC, and RBAC. In Proceedings of the 26th Annual Conference IFIP WG 11.23 Working Conference on Data and Applications Security and Privacy, pp. 41–65, 2012.
10. <https://en.wikipedia.org/wiki/Rule-based-machine-learning>.
11. <http://www.grymoire.com/Unix/Awk.html>.
12. Jha S., Li N., Tripunitara M., Wang Q., Winsborough W.:Towards formal verification of role-based access control policies. IEEE Transactions on Dependable and Secure Computing, pp. 242–255, 2008.
13. Sandhu R., Bhamidipati V., Munawar Q.:The ARBAC97 model for role-based administration of roles. ACM Transactions on Information and System Security, pp. 105–135, 1999

References

14. O. Ojagbule, H. Wimmer and R. J. Haddad, "Vulnerability Analysis of Content Management Systems to SQL Injection Using SQLMAP," SoutheastCon 2018, St. Petersburg, FL, USA, 2018, pp. 1-7, doi: 10.1109/SECON.2018.8479130.
15. Phillips C., Demurjian S., Ting T.C.: Safety and Liveness for an RBAC/MAC security model. In Proceedings of the Data and Applications Security XVII, pp. 316– 329, 2004.
16. Li N., Tripunitara M.V.: Security Analysis in Role-Based Access Control. ACM Transactions on Information and System Security, vol. 9, no. 4, pp. 391—420, 2006.
17. Rakkay H., Boucheneb Security Analysis of Role Based Access Control Models Using Colored Petri Nets and CPNtools. Transactions on Computational Science IV, pp. 149–176, 2009.
18. Ferrera A.L., Madhusudan P., Parlato Security Analysis of Role-Based Access Control through Program Verification. In Proceedings of the IEEE 25th Computer Security Foundations Symposium, pp., 2012.
19. Martin E., Tau X.: Inferring Access-Control Policy Properties via Machine Learning. In Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks, pp. , 2006.
20. Singh M.P., Sural S., Atluri V., Vaidya J., Yakub U.: Managing Multi-dimensional Multi-granular Security Policies Using Data Warehousing. In Proceedings of the International Conference on Network and System Security, pp. 221–235, 2015.
21. A survey of access control methods, National Institute of Standards and Technology, and National Security Agency. Technical report, 2009.
22. Jin X., Krishnan R., Sandhu R. unified attribute-based access control model covering DAC, MAC, and RBAC. In Proceedings of the 26th Annual Conference IFIP WG 11.23 Working Conference on Data and Applications Security and Privacy, pp. 41–65, 2012.
23. <https://en.wikipedia.org/wiki/Rule-based-machine-learning>.
24. <http://www.grymoire.com/Unix/Awk.html>.
25. Jha S., Li N., Tripunitara M., Wang Q., Winsborough W.: Towards formal verification of role-based access control policies. IEEE Transactions on Dependable and Secure Computing, pp. 242–255, 2008.
26. Sandhu R., Bhamidipati V., Munawer Q.: The ARBAC97 model for role-based administration of roles. ACM Transactions on Information and System Security, pp. 105–135, 1999

