**DETECTION AND PREVENTION OF**

**SQL INJECTION ATTACKS**

***B.TECH SEM – VII Mini PROJECT***

***Dept. of Computer Science & Engineering***

By

**Karansinh Vaghela 18BCP048**

**Rachit 18BCP089**

**Vyomesh Jethava 18BCP152D**

**Under the Supervision**

**Of**

**Dr. Payal Chaudhari**



**SCHOOL OF TECHNOLOGY**

**PANDIT DEENDAYAL ENERGY UNIVERSITY**

**GANDHINAGAR, GUJARAT, INDIA**

**July – December, 2021**

**ABSTRACT**

Personally identifiable information (PII) is information regarding things such as bank accounts, retirement or stock investments accounts, credit card accounts, medical records, or insurance claims. There is a need to protect the PII in databases that are connected to the ubiquitous, global network that is the Internet. If there is any vulnerability in the protection in a system that holds PII, then it presents an opportunity for an unauthorized person to access this PII. One of the techniques available to would be information thieve is SQL injection (SQLI). In this project, a system is developed to analyse the values submitted by users through HTML forms and look for possible attack patterns. Once the systems find such a pattern, it blocks the attack and makes a record of the activity. If an attacker continues to pass such attack patterns, the system blocks access by this user together. A mechanism is included to block users who attempt to log in at abnormally high rates. This provides a combination of pattern-based detection and anomaly-based detection to create a reasonably robust intrusion detection system, with respect to SQLI attacks.

**Table of Contents**

Introduction4

Literature Review5

System Model6

Proposed Work7

Implementation Details7

Results8

Existing Tools8

Conclusion and Future Direction9

**Introduction**

The OWASP(Open web application security project) is a worldwide non profit charitable organization that is continuously working on application software security issues and providing information about AppSec to corporations, individuals and organizations etc. to take decisions. The top 10 list consists of the 10 most seen application vulnerabilities. Among those vulnerabilities the top most one is SQL injection followed by broken authentication. SQL injection attack is a type of security vulnerability that target database connected web applications. It generally allows an attacker to view data that they are not normally able to retrieve In this attack, the attacker inserts a malicious SQL query into the web application to manipulate data or even to gain access to the back-end databases. This vulnerability mainly occurs due to weaknesses present in source codes. The other reasons for this vulnerability may be the weakness of the programming language or improper input validation. Most of theworks inDetecting SQLIAare focused on SQL injection structure at Application Level, but this method fails in detecting such attacks that use stored procedure and also the altered data that present in database system. The SQL injection attack can update data, delete data, insert and execute commands on server which cause to download and install Trojans and other malicious programs. Exporting valuable data such as credit card details, email, and passwords to the attackers remote server Getting user login details etc alter or delete data stored in the back-end databases, read sensitive information from the database and perform an administrative operation on the database for example shutdown of the database management system or sometimes an attacker attacks to compromise the underlying back-end infrastructure or server, or also can perform a denialof-service attack. A SQL injection attack is created by addition of a SQL inquiry with the input information from the client to the application. Many researches are working to find the vulnerabilities in web applications which are prone to the SQL injection attack and trying to provide solution to prevent them from happening. In this papermachine learning techniques namely SVM, Decision and Naive Based classifier and another Deep Learning classifier is implemented for detecting SQL injection attack. The performance of the techniques are tested and results compared with respect to Accuracy, Precision, using the synthesized Dataset obtained from Kaggle.

**Literature Review**

***George H John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In Proceedings of the Eleventh conference on Uncertainty in artificial intelligence, pages 338–345. Morgan Kaufmann Publishers Inc., 1995***

In this paper author proposed an algorithm to prevent the SQL injection attack at the login phase by combining both the code conversion and the parse tree validation methods. The author studied varies techniques for the SQL attacks and the prevention techniques. The parse method validates the user input for its vulnerability, and the code conversion takes place if there was a chance of vulnerability. The integrated algorithm is capable of preventing only text field SQL attack. Thus future work was to develop a better algorithm that can prevent the SQL attack through other method.

***Romil Rawat and Shailendra Kumar Shrivastav. Sql injection attack detection using svm. International Journal of Computer Applications, 42(13):1–4, 2012.***

Author had proposed two classifications namely Edit distance algorithm and Binary distance algorithm. The author compared the outcome of his proposal with support vector machine algorithm, Naive Bayes classifier and Parse Tree Based approach methods and justified his proposal was better. The limitation with this method was that it can handle only two types of SQL injection attacks whereas the literature claims that there are 6 types of attacks as listed.

|  |  |  |
| --- | --- | --- |
| Type | Description | Technique |
| Tautology | The condition of the statement gives always TRUE result | SELECT \* FROM tableName WHERE user login= or 1=1– |
| Union | Combining the results of two or more statements | SELECT \* FROM tableName WHERE user login= UNION SELECT \* FROM tableName WHERE No=12345 – AND passwd = AND pin= |
| Logical or illegally incorrect | Inject parameters which creates syntax, type conversion, or logical error | SELECT \* FROM tableName WHERE user login= ’kranthi”’ AND passwd = |
| Piggy Backed | Updateoriginal query by inserting additional queries to the original statement | SELECT \* FROM tableName WHERE user login=kranthi AND passwd=; drop tableName user – AND pin=221 |
| Stored Procedure | Run built-in functions using malicious SQL codes Modify the injection statement by alternating encoding to escape from detection | SELECT \* FROM tableName WHERE user login= ’kranthi’ AND passwd =’kumar’; SHUTDOWN;–; |
| Alternateencoding | Alter or modify SQL injection query statement to escape from detection | SELECT \* FROM tableName WHERE user login= ’kranthi’;exec(char(0x59842 352646f776e)) AND passwd =’kumar’ |

***Bharat Kumar Ahuja, Angshuman Jana, Ankit Swarnkar, and Raju Halder. On preventing sql injection attacks. In Advanced Computing and Systems for Security,***

Author had recommended the implementationof three different approaches to prevent the SQL injection attack namely Query Rewriting Approach, Encoding-based Approach and Assertion-based approach. Rewriting based approach is the easiest way to check whether SQL query is normal or an attack. But if the Query is longer one then computational overhead problem arises. The second technique is Encoding based approach technique. In this approach only UID and password was encoded and decoded.The limitation in this approach is, it has the code conversion overhead and it is time consuming method. The third technique is Assertion-based Approach. In this approach a piece of code (assertion) is added in the web application in order to verify that no SQLIA happens. The drawback of this approach is practically it is very difficult to implement and in case of large database it introduces computational overhead.

**System Model**

SQL Injected

SQL Query

SQL not Injected

Dataset

**PROPOSED WORK**

To overcome the limitations to detect the SQL injection attack we have simple and efficient method for SQL injection attack detection. Whenever a SQL query is generated from client side the proposed model classifies whether that query is injected one or normal query by using itspattern. The proposed model and the Machine Learning models were implemented using Python, and tested using the dataset from Kaggle.

**Methadology:**

Dataset Cleaning-Regular Expression library of python was used to replace numbers in the dataset with num.

Vectorization- Words, Special characters and numbers are converted to vectors for feature extraction using Sklearn library Count Vectorization technique.

Splitting of training and testing dataset using sklearn library.

Training dataset to learn patterns to detect SQL Injection. Algorithm used are Naïve Bayes, Decision Tree, Support Vector Machine, K-Nearest Neighbor, ANN and LSTM deep Learning models.

Finding performance, accuracy and precision of models and saving the best model.

Deployment of model using Flask API and designing web templates for flask.

**Implementation Details**

Project is implemented in

Pythton-3.8.12

Spyder IDE-4.2.5

Flask-1.1.2

Numpy-1.20.1

Pandas-1.2.4

Sklearn-0.24.1

Tensorflow-2.7

**Results**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** |
| ANN | 0.9311678267594741 | 0.9904610492845787 | 0.8824362606232294 |
| Naive bayes | 0.6728538283062645 | 0.6264521894548705 | 0.9929178470254958 |
| SVM | 0.757153905645785 | 1.0 | 0.5552407932011332 |
| KNN | 0.9102861562258314 | 0.9966329966329966 | 0.8385269121813032 |
| Decision Tree | 0.9242072699149265 | 0.995114006514658 | 0.8654390934844193 |

# **EXISTING TOOLS**

There are a number of existing tools available, both hardware and software based,to deal with SQL-Injection attacks. Tools exist to detect SQL-Injection attacks while others tryto identify and fix SQL-Injection vulnerabilities. The following are a few software oneswe will discuss.

• GreenSQL

• dotDefender

• CodeScan Labs: SQL-Injection

GreenSQL is a free Open Source database firewall that sits between the webserver and the database server and is used to protect databases from SQL injectionattacks. The logic is based on evaluation of SQL commands using a risk scoring matrixas well as blocking known database administrative commands (e.g., DROP, CREATE,etc). Reports are generated on timestamp, query pattern, reason blocked (e.g., trueexpression, has 'or' token). It has a white list of approved SQL patterns. However, onlyMySQL database is currently supported. In comparison, the IDPS in this project maybe used with any relational database, not just MySQL. The IDPS has both black andwhite list pattern features.

Applicure’s dotDefender is a web application firewall that offers a SQL-Injectionsolution. dotDefender is a multi-platform solution running on Apache and IIS webservers. Central management ensures a single point of control and reporting for allservers. There is an application layer firewall in front of web applications. It has a set ofsecurity rules that enable it to be a powerful solution. However, the cost is prohibitive.

The annual license costs $1,810 while a perpetual license is $3,995, which are both pricyfor personal use.While dotDefender is an expensive product, IDPS is a free product. Another product is CodeScan Labs’ SQL-Injection detection product. It has thecapability to scan web application source code that you selected for code syntaxvulnerabilities. It subsequently generates a "debug style" report. The speed depends onhow large the web application is and its complexity. The CodeScan software does not fixthe code, however; it only points out the issues. The company offers a 21-day free trial,but normally it requires a yearly subscription to be maintained. The actual price is notadvertised and one must contact sales representative to find out the cost. A separateactivation key is required for different programming languages and additional capabilities.

# **Conclusion and Future directions**

We have comparative study of SQL injection detection. Machine learning techniques namely SVM, Naive Bayes, KNN, Decision Tree and another deep Learning classifier was implemented and tested. The techniques were tested using Kaggle dataset. Finally we had deployed artificial neural network model and created the login page which prohibit SQL injection by deploying this model using Flask. This website shows alert box if user tries to inject malicious code in input fields of the webpage. Further we can implement a mechanism in which if an attacker continues to pass such attack patterns, the system blocks access by this user together. A mechanism can be included to block users who attempt to log in at abnormally high rates. This provides a combination of pattern-based detection and anomaly-based detection to create a reasonably robust intrusion detection system, with respect to SQLI attacks.