Tool-Based Prediction of SQL Injection Vulnerabilities and Attacks on Web Applications

B. Padmaja, G. Chandra Sekhar, Ch. V. Rama Padmaja, P. Chandana, and E. Krishna Rao Patro

Abstract In today's present time, SQL injection has become a significant security threat over the web for diverse dynamic web applications and websites. SQL Injection may be a sort of associate injection attack that produces it doable to execute malicious SQL statements into an online application consisting of SQL information. Attackers use these SQL Injection Queries or Statements specified if an Internet site or an application hosted on web contain SQL vulnerabilities to bypass application security measures. The Attacker will even go around authentication associated with authorization of an online page or Internet application and might bypass these methods and retrieve the content of the whole SQL information of an online application. The purpose of the proposed system is to predict the occurrence of a SQL injection attack on a particular server where an application is deployed from a given supply at a particular point in time. This predictive experiment is managed using the JMeter tool. From network logs, you can now pre-measure, exclude choices, analyze, and feed machine learning models to predict SQLIA.

Keywords SQL injection · Web application · Classification · Prediction · JMeter

1 Introduction

As the world is developing with the exponential development of Internet technology and therefore the development of recent Internet Products like Web 3.0, SNS, Apache Axis, the net-based web applications are being widely employed by all the people. At the same time evolving and rapid use of Internet-based Web applications and Web business attracted many hackers toward it which resulted in various Web security issues. Among all the online Security issues, one of the most notable and

B. Padmaja (✉) · Ch. V. Rama Padmaja · P. Chandana
Department of Computer Science and Engineering (AI & ML), Institute of Aeronautical Engineering, Hyderabad, India
e-mail: b.padmaja@gmail.com

G. C. Sekhar · E. Krishna Rao Patro
Department of Computer Science and Engineering, Institute of Aeronautical Engineering, Hyderabad, India

V. Bhateja et al. (eds.), *Communication, Software and Networks*, Lecture Notes in Networks and Systems 493, https://doi.org/10.1007/978-981-19-4990-6_49

dangerous ones is SQL Injection. SQL Injection ranks 6th in today's top 10 Security issues that are more dangerous, universal in Web application programs consistent with OPEN WEB APPLICATION SECURITY PROJECT (OWASP). Basically, the hackers tamper the Webpage contents and even steal the important internal information which is stored in the database by using SQL injection flaws. Even hackers install some malicious code into the database such that they can take over the entire system administration. For instance, in August 2009—Eleven Breach—a bunch of programmers (hackers) utilized SQL Injection to Penetrate corporate Systems at some organizations, principally the 7-Eleven corporate store, taking 130 million Mastercard numbers and their information. These SQL injection attacks have continued arising and therefore the SQL injection is presently till proceeding. In early 2021, 70 gigabytes of information were taken from the acute right website GAB by an SQL Injection attack. Therefore, it is very significant and important to review the prediction and prevention technology targeting SQL Injection attack to enhance the online security and to stop that attack before it is happening on a web Server. In this paper, the goal is to predict SQLIA in a web application. We used the open source software Apache JMeter to generate the log data. The log data is then preprocessed to perform a feature extraction and compare it to the available SQLIA search commands. This logistic regression model is used to predict the SQLIA in web Applications.

2 Procedure of SQL Injection

SQL Injection is especially done on Dynamic sites not on Static web content because dynamic websites load the data from an information base instead of stored on a system like Static website. SQL Injections are old, dangerous, powerful and versatile to attack on an online site database. The Procedure of SQL Injection is principally done in six steps mentioned in Fig. 1.

2.1 Seeking Injection Points

Injection point is the websites which contain injection flaws. There is no Injection flaw in Static websites of URL which are ending with HTML, etc. The Injection may only occur in Dynamic websites ending with '?'. This is often because the page information is loaded with different input queries from the users and so the net page is loaded by obtaining information from the database. It should be users' information or another site information. The SQL Injection is determined by the data given to the browser by inputting some special characters like 1 = 1 or)' within the address bar of the browser or the page table like login page. The SQL injections are performed through the Internet or mainly we will say Network. These attacks are versatile so much so that they will be performed from anywhere on Earth. Since the HTTP or HTTPS has two request methods GET and POST methods, SQL Injection

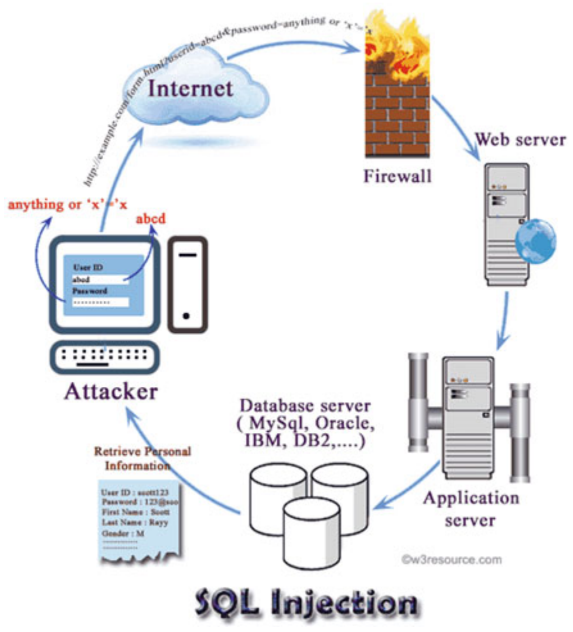


Fig. 1 Procedure of SQL injection

also has two methods GET INJECTION and POST INJECTION. The requests are very similar to Normal requests specified firewalls allow them as Normal HTTP or HTTPS requests hence no alarm is raised.

2.2 *Obtaining Information from Database*

Obtaining database Information is the main target in performing SQL Injection. The database type is judged by performing SQL false statements with special structures. As an example, there's URL LINK: http://www.amazon.com/show.asp?id=6. If we are going to add single quotation mark (') at the end of web link, i.e., is https://www.aamzon.com/show.asp?id=6', the type database will be judged to the result we get if the Microsoft server returns it an error, then the access may be judged as backend database login. When a user enters their valid username and password will pass authentication and the user will login.

SQL QUERY: SELECT * FROM USER_TABLE WHERE Username='user1' and Password='pass1'.

However, some users enter some malicious code in the Username and Password fields of the website where Username—usr1, Password—'or '1'='1.

The SQL QUERY now it will be constructed as.

SQL QUERY—SELECT * FROM USER_TABLE WHERE Username='user1' and Password=' 'or '1'='1".

Since 1 = 1 will always be true, this user will always login to the website. Now the user gets unauthorized access to someone else's Account details and this information is resulted within the serious consequences for the person whose account information had been stolen. This is a very Simple example SQL Injection attack for understanding, and most of the websites can easily prevent this type of attack. This is often a case of theft of data privacy. This will let an unauthorized user get access to a database system and may end up in various unauthorized actions on the information base like deleting, modifying, retrieving important information and many more dangerous things and an SQL Injection makes all this Possible.

Since 1 = 1 will always be true, this user will always login to the website. Now the user gets unauthorized access to someone else's Account details and this information can result in serious consequences for the person whose account information had been stolen. This is a very Simple example SQL Injection attack for understanding, and most of the websites can easily prevent this kind of attack. This is a case of theft and violation of data privacy. This will let an unauthorized user get access to a database system can result in various unauthorized actions on the database like deleting, modifying, retrieving important information and many more dangerous things and an SQL Injection makes all of this Possible.

3 Related Work

SQL injection attacks are difficult to get detected using the firewalls. These detections of those attacks are usually possible through IIS log, checking databases and user inputs. First, the IIS log file contains the pages accessed by users, their IP addresses and files they have accessed. With careful observation of log files, its size and content, one can detect the SQL injection attacks. Second in databases, if SQL injection attack is there, then it will generate a variety of temporary tables. A user can check those tables, its structure and content to determine whether an attack went on or not. Third, the user's input data are often checked by testing its format, type, length and range [1]. Kamtuo et al. [2] focuses on the relationship between each programming language and the normal delivery of web applications, as well as frequently reported vulnerabilities, and almost all SQLIA and XSS vulnerabilities are prevented by a simple mechanism [2]. Singh et al. [3] Described different types of SQL injection attacks: Fast Flux Sql injection, Compounded SQL injection, and Deep Blind SQL injection. The author also described various techniques for analyzing, detecting, and preventing these attacks [3]. Saqlain Hussain Shah (2018) [4], the author has summarized all known vulnerabilities and attacks. They outlined current technologies and approaches to server-side security for web applications. Devietal [5] conducted a survey on the tautology of SQL injection attacks, a technique widely used by banks on the Internet. In [6], the author predicted a method for validating input values in Java-based Internet applications that validate code instrumentation and run-time input validation for static computer storage devices. This approach looks for the target path or object constructor in the compiled Java category file and statically inserts the memory unit code module. Xiaowei et al. [6, 7] worked to protect online applications from various server-side attacks. The author described three commonly used approaches to check for vulnerabilities along with attacks, such as input validation, session management, and application logic. Current methods have been dealt with in two mechanisms: attacks and vulnerabilities on the one hand, and design goals and the Internet application phase on the other. Komiya et al. [8] discussed various varieties of vulnerabilities in input functions of web applications. The author used a variety of machine learning techniques like SVM, Naïve Bayes, KNN, etc. to detect vulnerabilities in web applications. Ross et al. [9] used two datasets one from the online application host and another from Datiphy appliance node associated with MySQL server to spot attacks. The author used rule-based learning algorithms, decision trees and neural network algorithms to spot the attacks with reasonable accuracy. Jemal et al. [10] discussed various SQL injection attacks, detection and its prevention. The author experimented on different attack sources and its types using machine learning and ontology. Das et al. [11] proposed multiple approaches for classifying a dynamic SQL query as sta or malicious. One is predicated on distance and therefore the other one is web profile-based. The author observed the supervised classification algorithms to be more efficient for identifying attacks and providing security. Som et al. [12] identified SQL injection attack (SQLIA) to be the most hazardous among all for web applications and therefore the author experimented on stored procedures to stop SQLIA. The author's work is to produce protection and prevent attacks on web pages from SQLIA. Medeiros et al. [7, 13] proposed a mechanism called SEPTIC for preventing attacks on a database. This method also assists in identifying vulnerabilities in many applications using MySQL. The author proved his technique to be more efficient than the prevailing solutions up to now. Lwin Khin et al. (2012) [14, 15] proposed a collection of static code attributes for predicting vulnerabilities in web applications like SQL injection and cross site scripting. The proposed vulnerability predictor attributes were more efficient and proven to detect quite 80% of the vulnerabilities compared to other techniques of that time. Padmaja et al. [16] proposed anomaly-based intrusion detection techniques using supervised machine learning models to detect malicious users in real time network traffic. The authors proved Artificial Neural Networks (ANN) to be more superior than Support Vector Machine (SVM) on NSL-KDD dataset [17].

4 Proposed Method

The SQL injection prediction of a particular system that the system is considered as a server and application system, and forecasting of SQL injection and predicting where SQL injection can be performed can be collected. From the IP address, clock, time and requests from requests requested by the user, requests from requests, requests and dates from the same IP address. Because the server is increasingly done in request, log data will increase with the information about confirmation of SQL injection prediction. The proposed system shown in Fig. 2 consists of system A, which can be

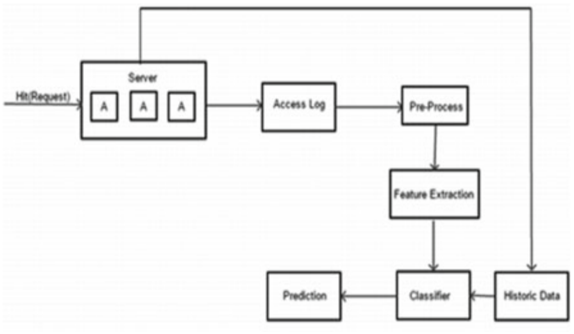


Fig. 2 Proposed method for predicting SQL injection attacks

provided from different servers or servers to another source to another source, and system A that can be provided at different times and different times It consists of To overcome this problem, the proposed system is all logistics from both servers, and data for predicting SQL injections is composed of all logistics from both servers.

5 Implementation

SQL Injection is a dangerous and old methodology which targets databases of web applications which also comprises the server as a normal request. The assault exploits weak input validation in code and site administration. The attacker can profit from the Database of web application when an SQL query is passed consisting of some characters which might comprise fire wall and http or https as Normal Query and may gain access to the administration user.

Http or https as Normal Query and may gain access to the administration user. As an example, observe the below PHP–1 code. If the user enters 'OR 1 OR username=' rather than $_GET ['username'] now it is an SQL Injection which might get access to the database. Hence the user can enter this query and can manipulate the database.

PHP–1

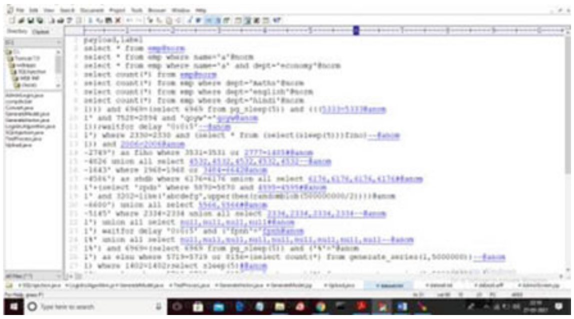
$result=MySQL query ('SELECT * FROM users WHERE username=" '. $_GET ['username'].'"');

If the user manipulates the get username the PHP–2 code is that the resultant which might be employed in login or search bar.

PHP–2

SELECT * FROM users WHERE username=" OR 1 OR username=".

Any requests sent to a web server are found in log data. This log data can contain may fields like IP address, date, Time, Request parameter etc. At first, we need to create a raw data which is then wont to feed to the logistic regression model. To make Log Data, we used a PHP login form. Here in order to create a huge data set we send numerous requests to the login page using Apache JMeter. Instead of sending requests manually one by one we can hit numerous requests by storing all requests like Username, Password in csv config and send them to the respective web application. This uses csv file parameterization for input without giving them manually. We are able to store all user login details and abnormal queries for username and password and might feed it to the JMeter. Now this dataset is employed in JMeter by adding a CSV config element in it such that we will hit the page with numerous requests with no problem. Now after processing all the information the log data is collected from JMeter which is stored to access log files from there it is converted into a csv file. From the csv file the IP address, date, time request parameter is extracted. Now the information within the request parameter is compared with the research SQL Commands and then convert them into SQL queries determining which are normal and which are abnormal queries Fig. 3 shows the dataset.txt which is employed for prediction. Now we upload the dataset which we extracted from JMeter and compare with research SQL queries for prediction of SQL Injection. After uploading the info now click on test preprocess and have selection and now the training model is going to be built and now click on generate prediction for the dataset Fig. 4 shows the train model.



```
payload,label
select * from emp#norm
select * from emp where name='a'#norm
select * from emp where name='a' and dept='economy'#norm
select count(*) from emp#norm
select count(*) from emp where dept='maths'#norm
select count(*) from emp where dept='english'#norm
select count(*) from emp where dept='hindi'#norm
1))) and 6969=(select 6969 from pg_sleep(5)) and (((5333=5333#anom
1" and 7528=2894 and "qoyw"="qoyw#anom
1));waitfor delay '0:0:5'--#anom
1") where 2330=2330 and (select * from (select(sleep(5)))fzmo)--#anom
1)) and 2006=2006#anom
-2769") as fiho where 3531=3531 or 2777=1485#anom
-4626 union all select 4532,4532,4532,4532,4532--#anom
-1643" where 1968=1968 or 3484=4442#anom
-4586') as xhdb where 4176=4176 union all select 4176,4176,4176,4176#anom
1'+(select 'rpdx' where 5870=5870 and 4595=4595#anom
1" and 3202=like('abcdefg',upper(hex(randomblob(500000000/2))))#anom
-6600") union all select 5564,5564#anom
-5145" where 2334=2334 union all select 2334,2334,2334,2334--#anom
1") union all select null,null,null,null#anom
1') waitfor delay '0:0:5' and ('fpnh'='fpnh#anom
1%" union all select null,null,null,null,null,null,null,null,null--#anom
1%') and 6969=(select 6969 from pg_sleep(5)) and ('%'='#anom
1") as elsu where 5719=5719 or 8156=(select count(*) from generate_series(1,5000000))--#anom
1) where 1402=1402;select sleep(5)#anom
```

Fig. 3 Dataset for prediction

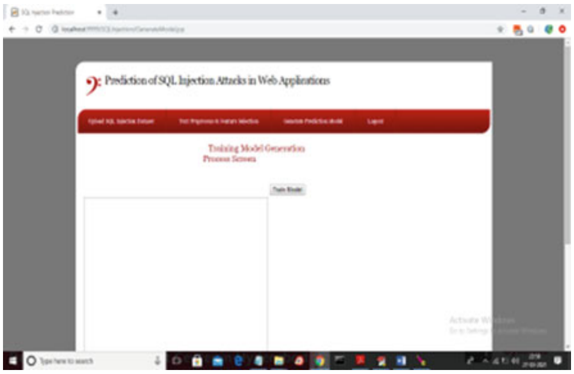

Prediction of SQL Injection Attacks in Web Applications
Upload SQL Injection Dataset
Test Preprocess & Feature Selection
Generate Prediction Model
Logout
Training Model Generation
Process Screen
Train Model

Fig. 4 Training the model

6 Results and Discussion

This work is used to predict SQL Injection which occurs on a web server or a web application from a given place at any particular time through the network. Here we use logistic regression for prediction, Logistic regression is basically a supervised classification algorithm. It is utilized for predicting the dependent variable by utilizing a given set of independent variables. Logistic regression can be worked with either yes or no 0 or 1, true or false etc. But instead of giving exact values it gives the probability value which lies between 0 and 1. Logistic regression is used to solve classification problems. Here we used the norm for normal queries and the anomaly for abnormal queries. You can use norm 0 and anomaly 1. Click Train Model to train your model using the uploaded dataset. Once the training model is generated, click Generate Predictive Model. The predictive output is now classified as a normal or abnormal query in all queries. The confusion matrix is a table that is used regularly to show the performance of the classification model. From the confusion matrix, the ROC score and the predicted percentage are calculated. This model is designed to anticipate SQL injection queries such as union, select everything, select where or as from like, and so on. Our models are trained on large datasets, 70% of which are used for training and 30% of which are used for testing. Confusion matrices and model predictions are based on considering only SQL queries that are classified as normal or abnormal. The model is deployed to a local server, where there is a SQL query that looks up the web address and predicts it as SQLIA or NOT. For logistic regression models, the dataset accuracy is 90%.

7 Conclusion and Future Scope

The main purpose of this research is to focus on a technique called SQL injection that allows hackers to execute malicious SQL queries on a database server. This is the most common technique used by hackers to retrieve information and data about what is stored in the database of online web-based applications. Here, the model proposed for predicting SQL injection attacks in web applications using JMeter can detect up to 90% of vulnerabilities. For future work, you can run the module to trigger an alert when a SQL injection attack occurs. Similarly, you can backtrack from the source IP to identify the actual area of the attack. You can also distinguish the actual client at the same time. You can run deep learning techniques and tree regressors for better results.

References

1. Qian L et al (2015) Research of SQL injection attack and prevention technology. In: 2015 IEEE International conference on estimation, detection and information fusion (ICEDIF 2015). https://doi.org/10.1109/ICEDIF.2015.7280212
2. Kamtuo K, Soomlek C (2016) Machine learning for SQL injection prevention on server-side scripting. In: 2016 IEEE international computer science and engineering conference (ICSEC), pp.1–6, Chiang Mai, Thailand
3. Singh JP (2016) Analysis of SQL injection detection techniques. Cryptography and Security, Cornell University. https://doi.org/10.20904/281-2037
4. https://github.com/SaqlainHussainShah/SQLi-Detection-using-Machine-Learning
5. Rubidha Devi D et al (2016) A study on SQL injection techniques. Int J Pharm Technol 8(4):22405–22415, ISSN: 0975-766X
6. Li X, Xue Y (2014) A survey on server-side approaches to securing web applications. ACM Comput Surveys 46(4), article no: 54, pp 1–29. https://dl.acm.org/doi/10.1145/2541315
7. Medeiros I, Neves NF, Correia M (2014) Automatic detection and correction of web application vulnerabilities using data mining to predict false positives. In: ACM 23rd International conference on world wide web, pp 63–73
8. Komiya R, Paik I, Hisada M (2011) Classification of malicious web code by machine learning. In: 2011 IEEE 3rd international conference on awareness science and technology (iCAST), China. https://doi.org/10.1109/ICAwST.2011.6163109
9. Ross K (2018) SQL injection detection using machine learning techniques and multiple data sources. Master's Thesis and Graduates Research, San Joe State University. https://doi.org/10.31979/etd.zknb-4z36
10. Jemal I, Cheikhrouhou O, Hamam H, Mahfoudhi A (2020) SQL injection attack detection and prevention techniques using machine learning. Int J Appl Eng Res 15(6):569–580, ISSN 0973-4562
11. Das D, Sharma U, Bhattacharyya DK (2019) Defeating SQL injection attack in authentication security: an experimental study. Int J Inform Security 18(3):1–22. https://doi.org/10.1007/s10207-017-0393-x
12. Som S, Sinha S, Kataria R (2016) Study on SQL injection attacks: mode, detection and prevention. Int J Eng Appl Sci Technol 1(8):23–29, ISSN No. 2455-2143
13. Padmaja B, Sai Sravan K, Krishna Rao Patro E, Chandra Sekhar G (2021) A system to automate the development of anomaly-based network intrusion detection model. In: 1st International conference on applied mathematics, modeling and simulation, journal of physics: conference series, vol 2089(1)
14. Padmaja B, Naga Shyam Bhargav P, Ganga Sagar H, Diwakar Nayak B, Bhushan Rao M (2021) Indian currency denomination recognition and fake currency identification. In: Journal of physics: conference series, vol 2089
15. Padmaja B, Srinidhi C, Sindhu K, Vanaja K, Deepika NM, Patro EK (2021) Early and accurate prediction of heart disease using machine learning model. Turkish J Comput Mathe Educ 12(6):4516–4528
16. Padmaja B, Reddy BR, Sagar RV, Pradhan HK, Chandra Sekhar G, Krishna Rao Patro E (2021) Prognosis of Vitamin D deficiency severity using SMOTE optimized machine learning models. Turkish J Comput Mathe Educ 12(6):4553–4567
17. Kishor Kumar Reddy C, Anisha PR, Shastry R, Ramana Murthy BV (2021) Comparative study on internet of things: enablers and constraints. Adv Intell Syst Comput