International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 05 | May - 2021 ISSN: 2582-3930

SQL Injection Prevention System Using Python

Mr. Rohit Chokhandre, Mr. Vaibhav Hadke, Mr. Adhaar Narnaware, Mr. Parag Rahate, Ass. Prof. Mr.ShrikantZade

Dept. of CSE,Priyadarshini Institute of Engineering and Technology, Nagpur
Dept. of CSE,Priyadarshini Institute of Engineering and Technology, Nagpur
Dept. of CSE,Priyadarshini Institute of Engineering and Technology, Nagpur
Dept. of CSE,Priyadarshini Institute of Engineering and Technology, Nagpur
Dept. Of CSE,Priyadarshini Institute of Engineering and Technology, Nagpur

Abstract - In today's world, SQL Injection is a majorInternet security threat for powerful web applications that live online. These Web applications perform many important processes for various web-based businesses. As the online use of various online services increases, so do the security threats on the web. There is a global need for all-powerful web applications and this general need is to store, retrieve or manage data in a database. Most data management systems and requirements such as MySQL Server and PostgreSQL use SQL as their language. SQL flexibility makes it a powerful language. It allows its users to ask what it wants without leaking any information about how the data will be downloaded. However, the widespread use of SQL-based data has made it a hotspot for hackers. They use Web applications with invalid codes to attack information. Introducing a visual SQL query, with unauthorized user input, in an official query statement. In this paper, we have tried to present a complete review of some types of SQL injection available, as well as the detection of such attacks and the security measures used.

Keywords–SQL, WEB, Injection Attack, Prevention Technology.

1. INTRODUCTION

In recent years, widespread adoption of the Internet has led to the rapid development of information technology. The Internet is used by the general public for purposes such as financial transactions, educational endeavors, and many other activities. The use of the Internet to perform important tasks, such as transferring the balance from a bank account, always poses a security risk. Today's websites strive to keep their users' data confidential and after years of running a secure online business, these companies have become experts in information security. The database systems behind these secure websites store sensitive information and sensitive information, in a way that allows information owners access faster while blocking login attempts from unauthorized users.

A common hacking technique is to try to get sensitive information from the database by first making a query that will cause the database parser to malfunction, followed by applying this query to the database you want. Such a way to gain access to confidential information is called SQL injection. As this information is ubiquitous and available online, dealing with SQL injection is becoming more important than ever. Although current data systems are less risky, the Computer Security Institute has found that each year about 50% of the database encounters a single security breach. The loss of revenue associated with such violations is estimated to be more than $4 million. In addition, a recent study conducted by the "Imperva Application 6 Defense Center" concluded that at least 92% of web applications could be infected with "malicious attacks".

To gain a better understanding of SQL injection, we need to better understand the types of communication that occur during a typical session between a user and a web application. The following figure shows the average interaction of all objects in a standard webapplication.

Fig. 1. SQL injection scheme

The web application, based on the above model, assumes text as input from users to retrieve data from the database. Some web applications think the input is legal and use it to create SQL queries to access the database. Since these web applications do not authorize user questions before sending them for details, they are vulnerable to attacks by SQL injection. For example, attackers, who pretend to be regular users, use a malicious text input with SQL commands to generate SQL queries at the end of the web database parser activates and releases sensitive information.

2. Body of Paper

I. TYPES OF SQL INJECTION ATTACK

In this project, we are dealing with the AND/OR and UNION-based SQL injection attacks.

• **AND/OR Attack:** One such attack is a basic attack that involves the OR concept in the SQL predicate. The hacker can specify malicious code such as ` OR 1=1 – on the form. The final query using this malicious code will be:

SELECT username, password FROM person WHERE username=`` OR 1=1 --` AND password=``;

Above malicious code ` OR 1=1 -- the first `(apostrophe) is used to close the string parameter now OR 1=1, what OR function does is it takes two input, if one of the inputs is TRUE, then OR returns TRUE. In our Example OR has two input one is string and other is 1=1 and 1=1 will always be TRUE so OR will also return TRUE and then query will also return TRUE as the WHERE clause will return TRUE. The -- (double hyphen) used in the code is used to comment the rest of the code, it will comment AND password=' '.

• **UNION Injection Attack**: The UNION Injection attack may be the most dangerous, but certainly the most surprising of the SQL-I attacks. This is because if it is successful, the UNION Injection attack allows the attacker to return records from another table. For example, an attack can modify the SQL query statement that you select in the user verification table to select another table as the shop table

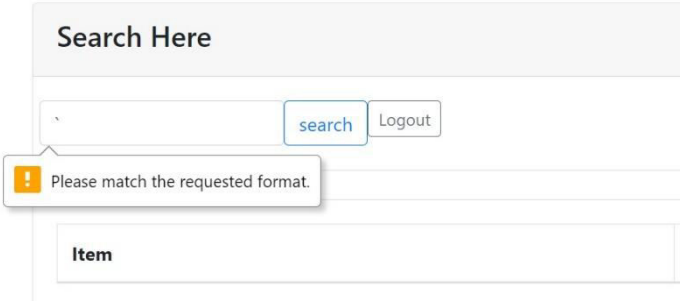
SELECT username, password
FROM person
UNION ALL SELECT item, price FROM shop

The use of UNION ALL in this attack allows the attacker to access the tables of the unstructured SQL query statement initially. The selected lines selected in both tables will appear on the resulting page.
The trick to this attack is that the columns selected in the second table must match the number (the same number of columns as the actual table should be selected) and then type. When you try to guess the correct number of columns, the attacker may simply continue to try to use a different number of columns in each attempt until he finds the correct number. To match the type, the attacker may try different types until they stumble on the right or they may choose to use NULL instead.

II. SQL INJECTION PREVENTIONTECHNIQUES

There are multiple techniques to prevent SQL Injection attack. In this paper we have demonstrate some of them.

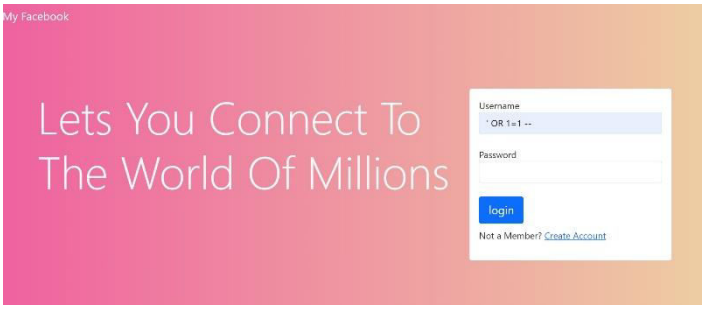
• **Cleanse and Validate user input:**
This is one of the most important and useful steps to prevent SQL Injection. Any type of data that user can provide, weather via a web-form, file, API use to be sanitized and the validate. This process will help us to check user input for invalid character, unacceptable length, or any other malicious code prior to processing or storing it on any production system.
This is the simplest step for the application UI to detect invalid characters and provide instant feedback.



Using this process, we can prevent the first step of SQL injection attack, because in the injection code there is "`" and this process will validate this symbol and will not allow any user to inject the malicious code it.

• **Stored procedures:**
Stored procedures is one of the technique to prevent SQL attack but this doesn't gives guarantee to prevent the attack successfully.
Stored procedures have same effect as that of parameterized query is implemented in same manner. They require the programmer to simply create SQL statements with parameters that are automatically added to the parameter without the programmer doing something particularly unusual. The difference between the prepared statements and the stored procedures is that the SQL code of the stored procedure is defined and stored in the database itself, and then called from the application. Both of these methods work in the same way in preventing SQL injection.
Note: 'Safe use' means that the database does not include any powerful SQL generation that is secure. Programmers rarely produce dynamic SQLwithin stored processes. However, it can be done, but it should be avoided.

• **Use of Prepared Statement (With parameterized query):**
Parameterized query makes it mandatory to the programmer to first define all the SQL code, and then pass in each parameter to the query later. This type of statementsis simple to write and easier to understand. Prepared statement allows the database to differentiate between code and data irrespective of what user input is supplied.
Prepared statement makes sure that an attacker is not able to change the intent of query.



In the above example if an attacker tries to enter the malicious code i.e., ' OR 1=1 --, the parameterized query would not be vulnerable and would instead look for a username which literally match the entire string ' OR 1=1 --.

Safe prepared statement example:
The following code example usus a prepared statement, python implementation of parameterized query.

```
sql = "SELECT * FROM person WHERE username=? AND password=?"
params = (username,password)
cursor.execute(sql,params)
```

3. CONCLUSION

SQL injection attack is a serious security problem in Web application system, most of which are caused by due to poor programming practices and attacker make use of such software defect to access the target databases**.**

This paper shows principle of SQL Injection, types of SQL attacks and prevention techniques.

ACKNOWLEDGEMENT

We would like to express a deep sense of gratitude to our project guide **Prof. Mr.Shrikant Zade** Department of Computer Science and Engineering of **Priyadarshini Institute of Engineering and Technology** for being the cornerstone of our project. It was his incessant motivation and guidance during the periods of doubts and uncertainties that have to help us to carry on with this project.

We would like to thanks **the Head OF Department Computer Science and Engineering** for providing the necessary guidance, support, motivation, and inspiration without which this project would not have been possible.

We would like to extend our special thanks to **Dr.Vivek Nanoti sir**, Principle of **Priyadarshini Institute of Engineering and Technology** for his encouragement and best wishes.

REFERENCES

1. Shen Qingni, Qingsi. Operating system security design. Beijing: Machinery Industry Press, 2013.
2. Yu Chaohui, Wang Changzheng, Zhao Yicheng. Practical Treasure Book of Network Security System Protection and Hacker Attack and Defense. Beijing: China Railway Publishing House, 2013.
3. Ma Limei, Wang Fangwei. Computer Network Security and Experimental Course, tsinghua university press,ISBN:9787302439332
4. Ma Limei,GuoQing,ZhangLinwei Ubuntu Linux operating system and Experimental Course, tsinghua university press,ISBN:9787302438236
5. Zhang shengcai,Zhoushuhui, SQL Injection Attack Prevention Technology Based on Improved Pattern Matching Algorithms, Technology Innovation, and Application,2017,35
6. Dong Zhenliang. Application of cryptographic algorithms and international standardization [D]. Financial Information Center of the People's Bank of China, 2018.
7. Zhou Yinqing, Ouyang Zichun. A brief discussion on the implementation and management of information system security level protection evaluation [D]. Digital Communication World, 2018.
8. Liang Lixin and Li Jun. Information Security Level Protection Evaluation Based on Virtualization [D]. Police Technology, 2014
9. Wubin, Liu Dun. SQL Injection Attack and Vulnerability Detection and Prevention Technology[D]. Network Security Technology & Application,201