International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Survey On SQL Injection Prevention and Detection

SHIRISHA N A*[*1],SHRAVANTHI H J[*2],VATHSALA N[*3],SUCHITH KUMAR G S[*4],CHAITRA P C [+5]

* Student,4th year,Dept of Computer Science Engineering,DSATM,Banglore
+ Assistentprofessor,Dept of Computer Science Engineering,DSATM,Banglore
1:shirisha.n.a11@gmail.com, 2:shravanthijayashankar@gmail.com 3:vathsalanagaraju4@gmail.com,4:suchithkumargs@gmail.com
5:chaitra-cs@dsatm.edu.in

ABSTRACT

With the growth of the Internet and the World Wide Web, human dependency on websites and web applications has increased significantly in the present day. Browsers and general web concepts are more familiar to most people to use than any other abstract computing interface. Users performing sensitive transactions online have paved a way for attackers to spoof and tamper the transaction data. SQL Injection is one of the most frequent web attack methods used by attackers to steal confidential data from organizations. It is a code injection technique that allows hackers to inject malicious SQL statements into input fields, which are then executed by the underlying SQL database. Attackers can also use web application URLs for inserting malicious SQL queries. Although a significant amount of research has been done on SQLIA Detection and Prevention, SQL injection attacks are still prevalent and cannot be eliminated. In this paper, we have provided an overview of SQL Injection vulnerability, different forms of SQL Injection Attacks, and threats posed by the attack. We propose a system to secure web applications using Hashing and Aho-Corasick pattern matching algorithm.

Index Terms—Website security, SQL Injection, String matching, Aho-Corasick, Hashing.

INTRODUCTION

Web applications have become an essential component in almost every sector including business, shopping, healthcare, etc. The global nature of the internet makes websites vulnerable to various attacks ranging from targeted database manipulation to large-scale network disruption. In the field of Web Application Security, SQL Injection is one of the most common attack vectors across the globe. It has still managed to be one of the OWASP top 10 web application vulnerabilities. The reason for this being SQL is still an extremely popular way to communicate with databases. It is a prominent method for performing various database operations like insertion, update, deletion, etc. Most of the web applications ranging from a simple website to a complex ecommerce website with millions of users are backed with databases. These databases contain private data and customer information. SQL queries are used almost every time a page loads. The attacker uses SQL Injection to compromise confidentiality, integrity, or availability of the database.

The security vulnerabilities generally arise as a result of programming code errors. A web application with thousands of lines of code makes it difficult for web developers to identify the code vulnerable to SQL Injection. The attacker takes the advantage of these flaws and injects a SQL query to gain unauthorized access to data in the database.Attackers can use input fields, website cookies, or URLs to perform SQL Injection.

Significant research work has been done on SQL Injection and various models have been introduced to prevent the attack. The diversity and large scope of SQL Injection make it difficult to ensure an adequate level of security against SQL Injection attacks. All the models could only prevent a subset of attack types. Filtering user inputs is one of the ways to prevent SQLIA. Even though filtering of user inputs provides a basic countermeasure, attackers can still manage to perform SQL injection attacks. The existing techniques pose different challenges and drawbacks. Few detection techniques produce false positives and false negatives while discovering the vulnerability. In this paper, we propose one of the efficient approaches for the problem of SQL injection that uses a combination of hashing and string matching algorithms which is effective against a larger subset of classic SQL Injection types.

RELATED WORK

A. SQL Injection Attack

SQL injection is a code injection technique for attacking data-driven web applications that involves inserting malicious SQL statements into an entry field for execution (e.g. to retrieve or modify the database contents). SQL injection attack allows the attacker to gain unauthorized access to the database and perform SQL operations like insert, update or even delete the data.

The root cause of SQL Injection Attacks in web applications is the lack of validation of user inputs received from the web form, input parameter, etc. Proper sanitization and filtering of user inputs like striping NULL characters, line breaks, single quotes, etc are some of the important countermeasures against SQL Injection. It should also be noted that sanitization of inputs is difficult and cannot be guaranteed to achieve 100% accuracy on every input.

B. SQL Injection Methodology

One of the causes for the SQL injection attack is that the user input is directly concatenated into the query. Since string concatenation doesn't care what you pass into it, it is a fast and simple way to construct SQL statements.

Considering a website that has

1). A homepage page listing all the products 2). When a user clicks the link from the homepage, the product ID is transferred to the get item page.

```
URL : /get item?id= 24753
```

The SQL query for this would be

```
"SELECT * FROM items WHERE id = " + id
```

The user entered ID value 24753 gets passed and a SQL query is built dynamically and hence the sql query variable will be

```
SELECT * FROM items WHERE id = 24753
```

The value of sql query is now passed into the database which retrieves the item and returns it, allowing the web page to be displayed.

The issue is that string concatenation is unconcerned with what you pass into it. It has no idea what an ID might look like. As a result, if a malicious user modifies the URLs ID value from "24753" to´

```
URL : /get item?id=(UPDATE items SET amount =
                                   0.1
                          WHERE ID = 24753)
```

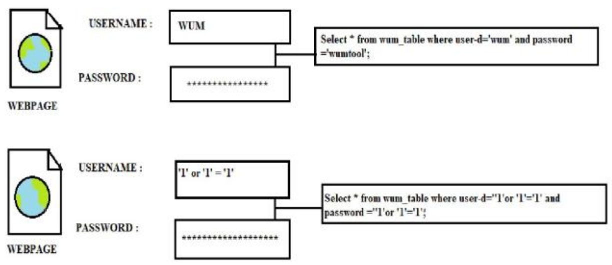
USERNAME :
WUM
PASSWORD :
****
WEBPAGE
Select * from wum_table where user-d='wum' and password ='wumtool';
USERNAME :
'1' or '1' = '1'
PASSWORD :
****
WEBPAGE
Select * from wum_table where user-d="1'or '1'='1' and password ="1'or '1'='1';

Fig.1. Displaysthedifferencebetweenqueriesgeneratedatthebackend beforeandafterSQLIA

The subquery that is substituted for the id of 24753 will run first and thus, the user can retrieve database records which he is not authorized to.

C. TypesofSQLInjection

SQLInjectionattackscanbebroadlyclassifiedinto3 categories:

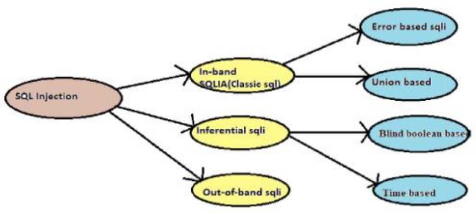

Fig. 2. Types of SQL Injection

1) In-band SQL Injection (Classic SQLi) : This type of SQL Injection is easy to exploit and hence most commonly used by the attackers. The attacker uses the same communication channel to both launch the attack and gather results.

Two sub-variations of this type are:

i) Error based SQLi

This technique relies on error messages received from the database server. Attacker enters invalid or unexpected input via user interface in order to receive error messages from the database. The received error message may contain details of the target to be attacked like operating system, version, structure etc. Using these details, the attacker can easily perform the attack on the target.

ii) Union based SQLi

This method uses SQL operator UNION that allows the attacker to run multiple SQL statements in a single query. Union operator can be used only if both the queries have exactly the same structure. Therefore, this attack can be performed only if the table name is known priorly. In addition, the number of columns selected and their data types in the original query statement must be known.

2) Inferential SQL Injection (Blind SQLi) : Blind SQL Injection relies on server response and behavior.

The attacker sends data payloads to observe response from the server. No data is actually transferred via web application and the attacker cannot see the result in-band. This type of attack is slow in general on large databases.

Two types of Inferential SQLi are:

i) Boolean based SQLi

The attacker sends a SQL query which forces the database server to return the result depending on whether the query evaluates to TRUE or FALSE. The attacker can know the result for the data payload sent, even though no data is returned from the server.

ii) Time based SQLi

This involves sending a SQL query to forcibly make the database server wait for a specified amount of time before sending the response. The attacker can infer if the query returned is TRUE or FALSE by observing the response time.

3) Out of band SQL Injection : This type of SQL Injection can be performed only if some features are enabled on the database server used by the web application and hence, it is not very common. The attacker cannot use the same channel to perform attack and gather results.

D. Threats posed by SQLIA

TABLE I

VARIOUS THREATS POSED BY SQL INJECTION ATTACKS WITH EXAMPLES.

SQLI TYPES	EXAMPLES
Spoofing	Use another user's credentials and retrieve data belonging to other users
Tampering	Modify data in the database, for example change other employee's salary
Repudiation	Delete transaction records or log files
Information disclosure	Access confidential data, for example credit card details
Denial of service	Run SQL queries that causes heavy load on server and brings the service down
Elevation of privileges	Gain administrator privileges by spoofing Administrator login credentials

LITERATURE SURVEY

[1],in this paper a hybrid string matching algorithm KMPS to detect SQL Injection, XSS, CSRF, and buffer overflow vulnerabilities. The algorithm uses shifting step from Sunday Search Algorithm and matching logic from KMP Algorithm to detect malicious pattern in a URL. The comparative study of proposed technique against Boyer Moore Algorithm shows better results in terms of searching time, throughput, and accuracy as shown in table II. Performance of the presented technique was examined by applying over 120 URLs. Authors conclude by highlighting that the vulnerabilities cannot be completely eliminated but different methods can be applied to detect the vulnerabilities at different stages of the web application development.

TABLE II

SHOWS THE RESULTS OF BOYER MOORE AND KMPS ALGORITHMS IN TERMS OF SEARCHING TIME, THROUGHPUT AND ACCURACY.

	Boyer Moore	KMPS
Seraching Time(ns)	264814477.6	127342594
Throughput	11.82879	20.84
Accuracy	70.9	72.8

[2],in this paper a technique to detect and prevent SQLIA using modified Aho-Corasik pattern matching algorithm, SQLMAP tool and AIIDA-Sql technique. The SQLMAP is a tool used to detect and exploit SQL Injection. The only drawback of SQLMAP is, it takes 15-20 minutes to detect SQL injected query. To overcome this drawback, authors use a new approach based on neural network called AIIDA-sql. After detecting infected query, it is then passed through static pattern list and compared for malicious patterns with use of Aho-corasick pattern matching algorithm. If the query is not present in the list, it will be accepted, otherwise rejected. The drawback of this system lies in training the neural network, which is an overhead in terms

of computation and this is crucial as the system accuracy is majorly dependent on this. The conclusion states that the system is efficient in space and able to achieve a fair accuracy rate.

[3],in this paper a technique called SQL Injection Protector for Authentication (SQLIPA). The technique focuses on prevention of tautology SQL injection attack. It uses the hashing mechanism for producing hash values of username and password input fields. Whenever a new user creates his account, hash values are generated for username and password field. These hash values are stored in separate columns in the database along with username and password columns. Whenever the users try to login, the login credentials are authenticated against the stored values in database and hashed values generated are checked against the stored hash values of username and password. Thus, adding a layer of security and preventing the tautology injection attack.

[4],in this paper a tool called AMNESIA for detection and prevention of SQLIA. The approach uses static analysis to identify hotspots and build a SQL-query model of legitimate queries and runtime monitoring to check query against pre-built model. The technique is automated and requires a web application as input and has no overhead of setting up any additional runtime environment. The empirical evaluation also shows its efficiency in detecting and preventing SQLIA. The drawback includes dependency of proposed method on Java libraries. This approach also produces false positives and false negatives in certain situations.

[5],in this paper the authors put forward a SQLIA detection and prevention technique using encryption with stored procedures. They propose the idea of generating a secret key in the middle tier for the user entered data and storing the encrypted data in the database.They use AES ENCRYPT() to generate hashes dynamically. With this implementation, the attacker cannot infect the web application because the secret key is unknown and generated dynamically.

[6],in this paper they have presented a comparative analysis of various types of string matching algorithms like Naive string matching algorithm, Brute force algorithm, Rabin-Karp algorithm, Boyer-Moore algorithm, KnuthMorris-Pratt algorithm, Aho-Corasick Algorithm and Commentz Walter algorithm. The performance, time complexity and space complexity are the considered metrics for the study conducted. The results of the analysis carried out are represented in the following table

TABLE III

SHOWS THE TIME COMPLEXITIES OF VARIOUS STRING MATCHING ALGORITHMS.

ALGORITHMS	TIME COMPLEXITY
Brute Force	O((n-m+1)m)
Rabin-Karp	$\theta(m),\theta(n+m)$
Boyer-Moore	$O(m+\|P\|), O(n)$
Knuth Morris Pratt	O(m),O(n+m)
Aho-Corasick	O(m),O(m+z)

CONCLUSION

SQLIA occur due to the flaws in the design of queries and an attacker takes advantage by inserting code in the query to modify the original query resulting in unauthorized access to database. This paper proposes SQLIA detection and prevention method that can be used as a countermeasure by integrating into any web application.Byrefering all the papers and literature survey we find Aho-Corasick algorithm as the optimal solution for the problem.

REFERENCES

[1] Komal, S.Deswal, “KMPS: A Hybrid Algorithm to Detect Web Application Vulnerabilities”, International Journal of Computer Sciences and Engineering Vol.-6, Issue-6, June 2018

[2] Nency Patel, Narendra Shekokar, “Implementation of pattern matching algorithm to defend SQLIA”, International Conference on Advanced Computing Technologies and Applications (ICACTA-2015), Procedia Computer Science 45 (2015) 453 – 459

[3] S. Ali, SK. Shahzad and H. Javed, “SQLIPA: An Authentication Mechanism against SQL Injection”, European Journal of Scientific Research, Volume.38, Number.4, pages: 604-611, 2009

[4] W. G. J. Halfond and A. Orso, ”Preventing SQL injection attacks using AMNESIA,” presented at the Proceedings of the 28th international conference on Software engineering (ICSE), ACM, Shanghai, China, Pages: 795-798, May 20–28, 2006, 2006.

[5] Deevi Radha Rani, B.Siva Kumar, L.Taraka Rama Rao, V.T.Sai Jagadish, M.Pradeep, ”Web Security by Preventing SQL Injection Using Encryption in Stored Procedures”, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (2), 2012,3689-3692

[6] Akhtar Rasool,Amrita Tiwari, Gunjan Singla,NilayKhare , “String Matching Methodologies:A Comparative Analysis”, International Journal of Computer Science and Information Technologies, Vol. 3 (2) , 2012,3394-3397

[7] Swapnil Kharche, Jagdish Patil, Kanchan Gohad, Bharti Ambetkar, Department of Computer Engineering Maharashtra Institute OF Technology,

Pune, “Preventing SQL Injection Attack using Pattern Matching Algorithm”

[8] Rua Mohamed Thiyab, Musab A.M.Ali, Farooq Basil, Abdulqader, “The Impact of SQL Injection Attacks on the security of databases”, Proceedings of the 6th International Conference on Computing and Informatics, ICOCI 2017

[9] B.J. Santhosh Kumar, KankanalaPujitha, “Web Application Vulnerability Detection Using Hybrid String Matching Algorithm”, International Journal of Engineering & Technology, 7 (3.6) (2018) 106-109