# **SQL INJECTION- Methods and Defense Mechanisms**

# Neeraj Ankam and Pradeep Bhat

University of Central Florida, USA

neeraj.ankam,bhatp93{@knights.ucf.edu}

**Abstract**

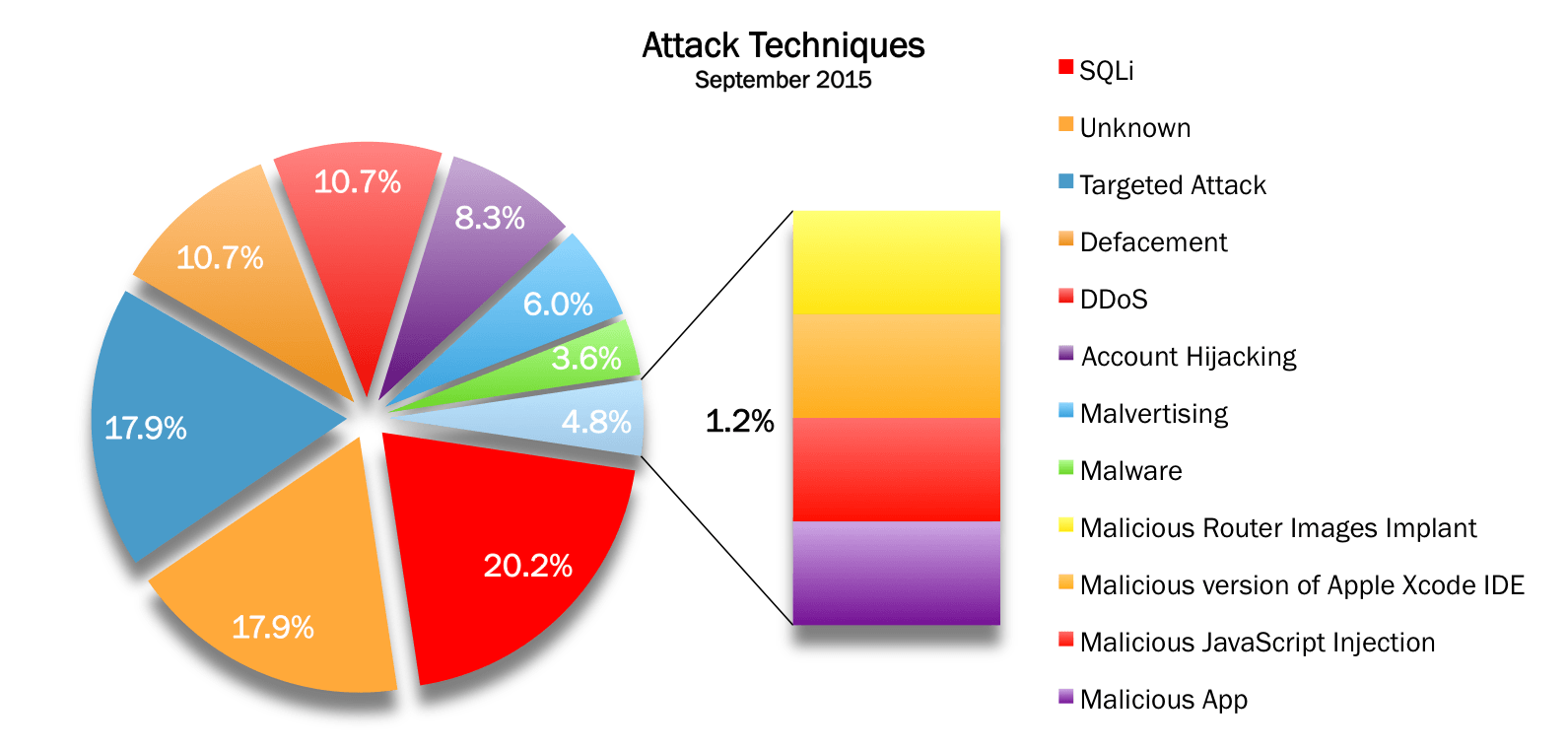
SQL Injection is a security exploit in which the attacker adds a Structured Query Language(SQL) query to the web form input box to gain access to the database or make changes to the code. The main consequences of such attacks include loss of confidentiality, authentication and lack of integrity as the hacker obtains access to the data of the database. In this paper, we discuss the several types of SQL injections, their effects and the diverse types of security techniques to safeguard the system against such attacks.

**1.Introduction**

Since the beginning of web programming, many companies started deploying their databases on the Internet for public use. These databases also contained sensitive and valuable information. SQL injection attacks are one of the most common vulnerability in the network where the hacker manipulates the data by using commands such as INSERT, UPDATE and SELECT according to his wants. The major consequences of SQL injection are mentioned below:

* Loss of Confidentiality: Because the hacker gains access to the sensitive information in the database [1].
* Loss of Authentication: As the hacker gains unauthorized access to the database without providing the credentials [1].
* Loss of Authorization: As attacker leaks the information present on the system [1].

The following diagram[8] shows that SQL Injection is one of the most widely done attacks.



In section 2 we discuss about related research work and motivation. In section 3 we have introduced SQL injection and described in brief about the threat they pose, types of SQL injection attack and the defense mechanism that can be used to prevent SQL injection. In section 4 we provide basic idea about steps taken to implement SQL injection. In section 5 we have discussed about the methodology we used to implement SQL injection. In section 6 we have elaborated one of the defense mechanism that can be used to prevent SQL injection which uses a framework called runtime monitors. In section 7 we have discussed about a technique to prevent SQL injection on stored procedure.

**2.Related Research Work and Motivation**

SQL Injection is a malicious technique where the hacker gains unauthorized access to the database through web application and modifies the data or uses it for his personal benefits. In [1] author enumerates the different type of attacks and talks about a few techniques that help prevent the attack. In [2] , the author presents an enhanced approach of Dependent on User Defined(DUD) by proposing an SQL sanitizer to detect attacks. In [3], the author briefly discusses about the various existing mechanisms to avoid the attack. In [4] the author describes a specific technique to prevent SQL injections in stored procedures. In [5], the author introduces a framework to detect and prevent union query based SQL Injection attacks.

**3.SQL Injection: Introduction**

Being one of the most widely used attacks, it allows the hacker to gain access to the database which contains sensitive information such as credit card details and other monetary information by manipulating the SQL query. The hacker could update or delete the data or use it for nefarious purposes [1].

1. Threats posed by SQL Injection

|  |  |  |
| --- | --- | --- |
| S.No | Threat | Description |
| 1. | Denial of Service | The server receives an inundation of requests and crashes temporarily due to which the user is denied access. |
| 2. | Destroys the existing data | After gaining access to the data, the attacker could erase the data in the database. |

|  |  |  |
| --- | --- | --- |
| 3. | Modify or Update data | The hacker could update or modify the existing data in the table according to his wish. |
| 4. | Changing the price of products | The attacker could change the prices of products / tickets and purchase them at lower prices |
| 5. | Gains access over sensitive information | The attacker could gain access to sensitive monetary information and use it wickedly. |

1. Types of SQL Injection attack

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | SQL Injection Types | Types | Description |
| 1. | Simple SQL Injection | a. Union SQL Injection  b. Error based SQL Injection | The Union based SQL injection takes information from the database in a legitimate way and appends its own query to obtain critical information.  Error-based SQL injection is an in-band SQL Injection technique that relies on error messages thrown by the database server to obtain information about the structure of the database. |

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | Blind SQL Injection | a. Boolean Based  b. Time based | It depends upon true or false values that we get from firing queries and get idea about the database.  It forces the database to delay the output for a particular amount of time if a certain condition is met. The hackers observe the pattern and carry out malicious activities. |

C. Defense Mechanisms

|  |  |  |
| --- | --- | --- |
| S.No | Name of the Mechanism | Description |
| 1. | Prepared statements  (with parametrized queries) | The coders should be writing queries such that they are bound with variables. This type of coding allows the developer to first define all the SQL code, and then pass the parameter to the query later. |

|  |  |  |
| --- | --- | --- |
| 2. | Constrain Input | By constraining the input used in your data access queries, you can protect your application from SQL injection. We need to validate all input to the application/website for type, length, format, and range so that undefined characters etc cannot be processed. |

**4. SQL Injection Step by Step**

The main steps involved in the attack are described below:

1)Information Gathering: The attacker first collects all the relevant information used to carry out the attack such as the database name et cetera.

2) Vulnerability Detection: The attacker checks if the database is vulnerable and it involves the sub steps:

a) First, he finds all the input fields.

b) Then he injects codes to generates errors.

c) One of the following characters is entered (‘), (;), (--) to check if the website is prone to attacks.

3) Launch the Attack: The attacker then launches the attack

4) Extract the data: After obtainment of access to the database, the attacker tinkers with the database and obtains sensitive information.

5) Interact with the Operating System

6) Compromise the system.

Let us take an authentic query to process the user data.

SELECT \* from Doctors where username= “xyz” and password=”1234”.

The above query is modified by the attacker in the following way:

SELECT \* from Doctors where username=” ” or 1=1 –‘and password=” ”.

From the above query 1=1 is always true and “—“comments the text[5]. With this query, the attacker could login into the application.

In almost all the web application stored procedures are used for the data communication assuming it reduces the possibility of SQL injection attack. But it is not real. A stored procedure, which is not effective or efficient, may cause the vulnerability to SQL injection attack. Attacker can use them to send the malicious query to attack the database [5].

**5. SQL Injection Implementation**

We have implemented SQL Injection by using Google dorks, SQL Map. Google dorks is a search string that uses advanced [search operators](http://whatis.techtarget.com/definition/search-operator) to find information that is not readily available on a website [6]. SQLMap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers [7].

We first used a google dork query to search for the list of vulnerable websites on Google.

inurl:index.php?id=

The execution gave us a list of the websites that had a similar structure to the query and which were potentially vulnerable.

We then checked for vulnerable sites by adding a (‘) symbol at the end of the URL and have found one site which was vulnerable(www.karaoke.co.nz). The error message was as follows:

You have an error in your SQL Syntax; check the manual that corresponds to your database version.

After ensuring the site was vulnerable, we used SQLMap to further check for vulnerabilities in the website and to collect information about the database.

python sqlmap.py -u

“www.karaoke.co.nz/items/index.php?id=37”

This query gave us the details of the database version used by the site. Following is the information we obtained regarding the database.

web application technology: Apache, PHP 5.2.17

back-end DBMS: MySQL >= 5.0

We then proceeded to find the databases using the query below

python sqlmap.py -u “www.karaoke.co.nz/items/index.php?id=37” -dbs

The query returned the names of the databases:

1.Claireg\_karaoke 2. Information\_schema

The following query gave us the list of tables present on the server:

python sqlmap.py -u "www.karaoke.co.nz/items/index.php?id=37" --tables -D claireg\_karaoke

This returned the results: 41 tables found.

We then extracted the details of a few users by executing the following query.

python sqlmap.py -u "www.karaoke.co.nz/items/index.php?id=37" --dump -D claireg\_karaoke -T Person

We have thus successfully managed to gain unauthorized access to the database and extract the details.

**6. Runtime Monitors to prevent SQL injection attack.**

A unique idea was proposed in the paper[4]. This paper focuses on keeping a track on the query traffic after deployment of web application to prevent union based SQL injection. A framework called runtime monitor is used for this task and its primary goal is to tackle union based SQL injection. The union operator can obtain information from multiple tables, compromising critical data from important tables in the database. The query used can be as follows

Select \* from table1 conditional operator UNION select \* from table 2 - - And operator

This type of query can be used to authenticate user with valid login ID and password. The conditional operator would check for the username and the AND operator to verify the password. The – operator is used for comments. The AND operator would be neglected because it is being used after the comment operator. The first SELECT query would be to validate username which would be incorrect but the second SELECT query would be executed and the malicious user would get access to multiple tables.

To keep a check on the invalid usage of SQL queries the authors have proposed the usage of a model called runtime monitoring framework. The framework is divided into three sections

1. Critical Variables Identification(CVI).
2. Path Identification Function(PIF).
3. Monitor Development and Integration (MDI)

The CVI, as the name suggests would identify all the critical variables in the system. These critical variables are the ones that store input information from the user.

The PIF would check for the path taken by these critical variables. The output from this module will be fed as input to MDI.

The MDI monitors the path taken by the critical variables and if it finds any abnormal behavior it notifies the administrator about the aberrant behavior and stops the flow.

The research team developed a website using simple HTML and CSS and used MYSQL as the database. The research team was successful in tracking all union based SQL injection.

**7. Preventing SQL injection attack in stored procedure.**

SQL Injection Attack is one of the common attacks where the attacker penetrates in the database layer through application layer and exploits the confidential or the financial data stored in the database. Therefore the web application hosted on the Internet or the Intranet should implement all the security measures needed to defend the attack and keep the data secure.

Many of them suggest using Stored Procedure to avoid SQL Injection Attack because the definition of the tables, database structure remains hidden even at the Database Access Layer.

In this paper[5] the author shows that even Stored Procedures are vulnerable to SQL Injection Attack and proposed a technique to detect SQL injection Attack in Stored Procedure. The structure of the original SQL Statement is compared with SQL statement with user input to check the difference between SQL Statement structures.

The user inputs containing the SQL keywords or logic leads to alter the original statement of the procedure which can change the field length or exploits the data. The user inputs should be validated or sanitized at the application layer itself so that SQL Injection Attack can be avoided.

There are various works related to SQL Injection attack defense at the application layer using some checks at the input field level, using SQLrand, using runtime validation or the combination of static and runtime validation.

The authors proposed a technique which uses combination of static and runtime validation to detect SQL Injection Attack at Stored Procedure level. In this process all executed SQL Statements are labelled in control flow graph and a SQL graph is generated. The user depended SQL Statement is recognized from the graph and monitored at the runtime to detect any alteration in the SQL Statement structure using Finite State Automaton.

The SQL graph is built by labelling all the SQL Statements which gets executed and the user inputs used in the procedure. The dependency between the SQL queries is plotted if the user input of one query is superset for another query. The relationship between user input and query is plotted when the user input is utilized in the query. From this graph the user input dependent query is recognized and selected for runtime validation. The use of SQL graph reduces the runtime overhead and maintains the efficiency of the query.

The runtime validation is done by the function SQLIAchecker which is called before the query execution is called. In this function the query structure before the user input is compared with the query after the user input, this comparison is made possible by appending a mark function before user input in query which return a random string or can be a session ID.

In SQLAchecker function the Finite Sate Automaton is first build with the query excluding the user input then this structure is used to compare with the Finite State Automaton build with the query including user input, if the structure of the query deviates from the original structure then a flag is set for SQL injection attack and the query is not executed.

The authors made a prototype of their proposed technique and carried out the experiment to test the performance based on false positive and false negative and to check runtime overhead cause due to runtime validation. They tested various SQL Injection attack like inserting tautology, inserting other SQL statement etc. and all the SQL Injection attack were detected and also found out that the runtime overhead cause due to building Finite State Automaton and validating queries is negligible compare to unsecure query executed. Thus, the proposed technique can be implemented in the web application to make the Stored Procedure calls secured.

**8.Conclusion**

SQL Injection attacks are one of the most common attacks and the attacker could gain unauthorized access to the databases.

In this paper, we have discussed about SQL Injection, the various threats it poses, the types of SQL injection and defense mechanisms. We have successfully demonstrated how SQL Injection takes place. We believe that each technique has its own downfall and hence in order to keep a website safe, a combination of various techniques should be used.

REFERENCES:

1. N. Singh, M. Dayal, R. S. Raw, S. Kumar. SQL injection: Types, methodology, attack queries and prevention. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2016:2872-2876.
2. R. P. Karuparthi, B. Zhou. Enhanced approach to detection of SQL injection attack. *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2016:466-469.
3. Amirtahmasebi K, Jalalinia SR, Khadem S. A survey of SQL injection defense mechanisms. . 2009:8 pp.
4. R. Dharam, S. G. Shiva. Runtime monitors to detect and prevent union query based SQL injection attacks. *2013 10th International Conference on Information Technology: New Generations*. 2013:357-362.
5. K. Wei, M. Muthuprasanna, Suraj Kothari. Preventing SQL injection attacks in stored procedures. Australian Software Engineering Conference (ASWEC'06). 2006:8 pp.
6. http://searchsecurity.techtarget.com/tutorial/SQL-injection-protection-A-guide-on-how-to-prevent-and-stop-attacks
7. www.sqlmap.org
8. http://opensources.info/september-2015-cyber-attacks-statistics