SQL Injection Attacks and Defensive Techniques

Dr. RajashreeShettar1, Agneev Ghosh2, Amal Mohan3, Amith Pramod4, Chetan Raikar5
Department of Computer Science and Engineering,
R.V. College of Engineering, Bangalore, Karnataka, India

1rajashreeshettar@rvce.edu.in, 2agneev_xaviers91@yahoo.com, 3amal.phelsea@gmail.com, 4amithpramod@gmail.com, 5chetanraikar12@gmail.com

Abstract

SQL injection is one of the top threats to any web application which interacts with a database system. It basically takes advantage of the loopholes in the server side program which interacts with the database server. SQL injection is one of the highly dangerous threats because it is easy to generate, difficult to design a defence mechanism and the data vulnerable to this type of attack is highly sensitive such as passwords, credit card details, etc. This paper compares different mechanisms used to defend against SQL injection attacks. Syntax of all the examples depicted in this paper is based on MySQL database system.

1. Introduction

Web applications play a vital role in today’s world. With the increased deployment of such web applications there has been an increase in the number of attacks targeting such applications. According to OWASP (Open Web Application Security Project) 90% of web based vulnerability is due to input data which are faulty and application does not have mechanisms to filter the input [1].

Web applications which interact with database as depicted in figure 1 are used nearly in all information systems and business applications like e-commerce, banking, transportation, web mail, blogs, etc. Web applications typically interact with backend database to insert and retrieve persistent data. These applications are so exposed to attacks that any existing security vulnerability will be exploited, which will lead to high negative impact on users. The major challenge faced by all the organization is to protect their precious data against malicious access or corruptions.

SQL(Sequential Query Language) injection attacks are application layer attacks which represent a serious threat to any database driven web application and they are one of the most frequent web attack. The web applications that are vulnerable to SQL injection may allow an attacker to gain complete access to the database. In some cases, attacker can use SQL injection attack to take control and corrupt the database system that the web application uses [2] [3].

SQL injection refer to a class of code–injection attacks in which data provided by the user is included in an SQL query in such a way that part of the user’s input is treated as SQL code [3]. SQL injection is carried out by including portions of SQL statements in a web application entry field in an attempt to get the website to execute the newly formed illegal SQL command on the database. According to OWASP report for 2010 and 2013 years top ten web application vulnerabilities are of SQL injection type which is the most common and the most exploited vulnerability [4].

2. SQL injection attacks

One of the main causes of SQL injection vulnerability is bad programming and inadequate validation of user input [5]. This vulnerability is exploited by attackers using different types of SQL injection attacks. The attackers usually inject malicious queries into HTTP requests [6]. Although there are infinite possible ways of framing an attack, they can be classified under seven types [6][7]:

- Tautology Attack
- Piggy-Backed Queries
- Union Query
- Illegal/Logically Incorrect Queries
- Stored Procedures
- Alternate Encodings
- Inference

Example signatures of different types of SQL injection attack that can be used to attack a vulnerable MySQL database is shown below.
Table-1. Example signatures of SQL injection attacks

<table>
<thead>
<tr>
<th>Attack</th>
<th>Form input example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tautology</td>
<td>'or 1=1'</td>
</tr>
<tr>
<td>Piggy-Backed Queries</td>
<td>; drop table student--</td>
</tr>
<tr>
<td>Union Query</td>
<td>' union select pin from bank where accno=145872 --</td>
</tr>
<tr>
<td>Illegal/Logically Incorrect Queries</td>
<td>'</td>
</tr>
<tr>
<td>Stored Procedures</td>
<td>EXEC sectionA') UNION ALL (SELECT passwd FROM staff</td>
</tr>
<tr>
<td>Alternate Encodings</td>
<td>'Exec(Char(0x73687574646f776e))</td>
</tr>
<tr>
<td>Inference Attacks</td>
<td>Blind injection: 'or 1=1', 'or 1=2' and so on till required information is collected.</td>
</tr>
<tr>
<td></td>
<td>Timing injection: ' and substring (select top 1 name from sysobjects,1,1) = A waitfor5--</td>
</tr>
</tbody>
</table>

2.1. Tautology attack

Tautology attacks are the type of attack in which the attacker injects SQL query snippets into conditional statements which will be evaluated to true. Tautology attack is the most common SQL injection attack. Table 1 depicts the signature of tautology attack.

2.2. Piggy-Backed attack

Piggy-back means “a ride on someone's back and shoulders”. Similarly, piggy-backed query SQL injection attack is a type of attack in which the original intended query to be executed is not modified; instead a new query is inserted with the existing query [7]. As a result the database executes the intended query and the injected query which can cause harm to the database. This type of attack depends on the configuration of the database depending on whether it allows multiple statements to be executed. A simple piggy back query attack signature is shown in table 1 (deletes table student if it exists in the schema).

2.3. Union query attack

Union query is a type of attack in which the union keyword is used. The dataset that is returned is a union of the original query and the query injected. Union Query attack is one of the most dangerous SQL injection attack as keywords like ‘load_file’ can be used with it to load system files of the system under attack [8]. Table-1 shows a sample signature of union query based SQL injection attack. The database takes the results of original intended query and performs union operation with the query in the example, which forces the target database to return a dataset with sensitive information (in the example pin of account number 145872 is also returned) [7].

2.4. Illegal/Logically incorrect query attack

Attackers use this approach to gather important information about the type of database and its structure [7]. Attacks of this nature are often used in the initial phase of an elaborate attack to gain information about the target database. Error pages returned by the database that are not filtered by the developer can contain important information. The error messages returned are of three types-syntact, type conversion, or logical error. Even if the application sanitizes error messages, the fact that an error is returned or not returned can reveal vulnerable or injectable parameters [7][9]. Table-1 shows the simplest method to generate error message, by addition of an extra single quote which can generate an error message.

2.5. Stored procedures attack

Most of the databases in use have in-built stored procedures. Stored procedures are also used by developers to perform some specific function. Stored procedures are used as a mechanism to counteract SQL injection. But stored procedures which use user inputs as its parameter is vulnerable to SQL injection attack [10]. For instance, if a database system has a stored procedure section, and if it accepts one user input as parameter it can be exploited by using the example in table-1. The stored procedure is possible only if the attacker knows the underlying database syntax.

2.6. Alternate encoding attack

In alternate encoding attack, the attacker attacks the target system using any of the other attacking techniques but the data is encoded to avoid detection. Alternate encoding, by itself is not an attack, it helps other attacks to evade the defence mechanism installed. As there are a large number of encodings possible it is difficult to develop a system which detects alternate encodings effectively. Table-1 shows hexadecimal encoded snippet equivalent to shutdown command which shuts the target database down [11].

2.7. Inference attack

In inference SQL injection attack the attacker tries to execute different types of queries on the target database and tries to infer details about the application based on the type of response from the application. This type of attack is usually used on applications which are not vulnerable to illegal/logically incorrect query attacks. There are two types of inference attacks:
a. **Blind injection:** In this type of attack, the attacker executes queries whose outcome is unknown to the attacker and infers the details of the database from the result of the query. In the example of table-1 the attacker tries various combinations of tautologies to infer details about the application [7].

b. **Timing attacks:** In this type of attack the attacker executes queries whose outcome is unknown to the attacker and infers the details of the database from the timing delays of response of the database to the query. In the example of table-1 the attacker tries to deduce the name of table in the database by introducing a delay if a character is matched. This is continued till the name of entire table is acquired [7].

3. SQL injection attack detection and prevention techniques

In order to accomplish the elusive task of shielding a Web application from SQL Injection attacks, there are two major issues which must be considered with great solemnity. Firstly, there is an imperative need of a mechanism to detect and precisely identify SQL Injection attacks. Secondly, knowledge of SQL Injection Vulnerabilities (SQLIVs) is crucial for securing a Web application. Researchers and developers over the years have suggested a few practices which have been proved to be quite reliable and prevents the developers from the demons of SQLI. One of the rudimentary approaches adopted by most developers is the use of prudent and vigilant coding practises. The root of the problem lies in the inefficient validation of the input. Thus an obvious solution to the problem would be to implement defensive coding practises [7]. Some of the most effectual coding practises which have been proved to be quite reliable have been summarized below:

a. **Input Explication:** Input provided to web applications is the primary pathway for malicious queries to infiltrate the network firewalls and other preventive setups. To exemplify, often the entire input is considered as string, often overlooking the usage of numbers. Ideally, while accepting numbers as inputs, any input apart from numbers must be ignored.

b. **Positive pattern matching:** Under this coding scheme, instead of identifying harmful and bad data, good and admissible data is identified. This practise makes the task of the developer comparatively simpler as attack patterns can have infinite possibilities in their structure which would be make its analysis highly demanding. Thus, by using only good data, the contents need to be matched with a static rule list thereby easing out the work of the developer.

However, the above mentioned technique has considerable shortcomings which can be exploited by adroit hackers in order to bypass the preventive measure and inject malicious queries into the protected network. Thus to overcome these restrictions, more advanced and reliable techniques, to detect and subsequently prevent SQL Injections must be taken into consideration.

**Information-theoretic approach:** In this approach, to detect SQLI attacks, the authors have a view point that the complexity of the intended query is compromised by the attack query. This framework has 2 phases; Training phase and Detection phase. In Training phase, the entropy of the all the queries present in the program are computed before the program deployment. In Detection phase, during program execution, the entropy of the invoked SQL Query is again computed. In this approach, it is assumed that there will be significant variations in the entropies of the invoked query and the original query if there is there is an attack. This variation is used to detect SQLI attacks. This approach is effective for a set of vulnerable programs implemented in PHP [13].

Xi-Rong Wu and Patrick P.K Chan proposed a SQLI detection technique in adversarial environments by **K-Centres.** This method is used to detect SQLI attacks before the SQL queries are sent to a database server. In an adversarial environment, the different types of attacks changes after a period of time, so the number and the centres of the clusters are adjusted according to the unseen SQL queries. The SQL query type, SQL keyword occurrence frequencies, SQL query length, time of the user input are extracted as features and the nearest centre distance is computed by Euclidian Distance. This method classifies a SQL statement as normal only if the calculated distance is less than a threshold and the length of the feature is less than that of the nearest centre.

The advantage of this method is that it’s satisfying performance under adversarial environment. Also this method is not sensitive to the ratio of normal and attack statements. The main drawback of our method is that it must receive a true label of each statement after classifying [14].

**Tokenization:** In this technique, the authors have proposed a method to detect SQL injection attacks by using query tokenization that is implemented by the Query Parser method. This technique checks whether the query generated on user’s input changes its intended result. This approach detects a single quote, space or double dashes in a query and each token consists of the strings before a single quote or a space or double dashes. This tokenization technique is
applied for both original query and the query with the injections. All tokens of a query are grouped into an array. Both the arrays are compared and if they differ in length then a SQLI threat is detected [15].

**Static and dynamic analysis** is an amalgamation of static and dynamic methods to detect SQLI attacks. AMNESIA (Analysis for Monitoring and Neutralizing SQL Injection Attacks) uses a model-based approach that combines static analysis and runtime monitoring [16] [17]. In the static phase, AMNESIA uses static analysis to build models of the different types of queries an application can legally generate at each point of access to the database. It uses a repository of all possible safe paths for its analytical purpose. In its dynamic phase, AMNESIA captures all queries before they are forwarded to the database and checks each query against the statically built models. Any queries which do not abide by the rules as rendered by the statically built models are tagged malicious and are prevented from being executed on the database.

**Pattern matching algorithm:** For threat detection an efficient pattern matching algorithm can be used for detection. Pattern matching is a technique which can be used to identify or detect any anomaly packet from a sequence of related actions [2]. Injection attack is a method that can efficaciously inject any kind of malign or anomalous string on the original string. Pattern based techniques are generally used for static analysis and patterns are generated from the attacked statements. Specifically the pattern matching algorithm can be built along the lines of the Aho-Corasick [2] pattern matching algorithm. The above algorithm generates a NFA and consequently uses two pointers, one of which points to the beginning of the pattern and the other to the denouement of the pattern and subsequently perform the pattern matching.

The pattern matching algorithm can be realised for a Signature based Intrusion Detection System which analyzes packets in the network and compares them against a pre-defined list of attack patterns known as signatures. However, signature based solution alone is not sufficient to defeat SQL injection attacks which is essentially due to the power of SQL and the flexibility that it gives to the user [18].

4. Conclusion

SQL injection has a deleterious impact on web applications, which in turn will have an inimical impact on online business. Up until now many of the approaches have been proposed by developers, with the purpose of defeating SQL injection attack both during coding and execution phase. One of the solutions to this is by using Intrusion Detection Systems. Among all types of IDSs (Intrusion Detection Systems), Signature based IDS have been the popular choice for most network administrators for their sheer reliability and accuracy. On the flip side, due to the static nature of signature based IDSs they fail to detect and identify any new attack which do not correspond to any of the existing attacks in the static rule list, and thus the attacker can bypass the detection by changing the appearance of the attack. In this paper, most of the popular existing SQL injections related issues have been reviewed. Key findings of this study could be summarized as:

a. A brief background of SQLI attacks pertaining to its way of functioning and its relevance in the network security world.

b. Detailed survey report on various types of SQL Injection attacks, vulnerabilities, detection, and prevention techniques

c. A comparative analysis of techniques based on their performance and practicality.

Thus as the current situation very bluntly stands, despite secure programming practices being in place and many proactive countermeasures being available, most of the web-based programs still suffer from SQLI vulnerabilities. Exploitations of such nature result in unwanted program behaviours and loss and leakage of confidential information of end users. Thus, curbing the ill-effects of SQLI has become highly imperative in order to ensure a hassle-free internet experience.

5. References


