Database Security —— SQL Injection

Fengxiang Lan
flan@ur.rochester.edu

Jing Zhang
jzh136@ur.rochester.edu

ABSTRACT
In this paper, we overview the database security, and introduce the SQL Injection Attack. In addition, we summarize the state of art and trend of researches in the field of SQL Injection Detection methods.

1. INTRODUCTION
Since DBMS (Database Management System) was published by General ElectricCo. in 1961, this system has become a crucial part of the real world. By utilizing databases, we can store, read and use millions of data at same time. As the organizations increase their utilization of database systems to handle the day-to-day tasks, to ensure the security of database become more and more important. With the growth of customer scale and service type, it become more and more difficult to ensure the legitimate connection. The arrival of the Age of Internet brings more challenges for maintaining security of database. In the Internet Age, one of the most common attack means to database system is SQL injection. This attack is so common that almost all websites have ever encountered it’s also so flexible that it’s hard to discover and avoid it. In 1998, the concept of SQL injection was put forward by Rain Forest Puppy for the first time. Since that, millions of researchers have paid a lot of attention on this field. In this paper, we will summarize the main methods to prevent this attack from happening and introduce the latest research achievements in this field. The following is the structure of this paper.

In the second part, we will take an overview of database security, which will include not only the traditional database security problems but also the latest users are facing in this Internet Age. Since the SQL injection is non-negligible for maintaining database security, especially security of network database, we will state this problem in detail in part 3. Finally, in the last part, we will present the researches about preventing SQL injection.

2. Database Security
Database provides a method to store, exchange and modify data more effectively, which was proposed in 1960s, because of the popularity of computer system. Once the database system was used among multiple areas, the security of database gained wide attention from the researchers. In 1970, S Jajodia and S Toy stated the concept of database security clearly in their paper [1]. Database Security, just as the name suggests, it means protecting database from losing data, modifying data illegally and using data improperly. [2]

Today the database can be used in various fields, such as medical, economic, education, military, administration. These databases contain pretty important and confidential information so there is a chance of attacks [3]. Even though database system faced with many different kinds of attacks, these database security problems can be summarized into the following categories [4]

1) Inference

By this way, hackers do not need to steal the sensitive data directly, they only need to derive the useful information from non-sensitive data [5]. Usually, hackers use these three methods to carry out this attack: Indirect attack, tracker attack and liner system vulnerability. They all get sensitive data with mathematical method.

2) Passive Attacks
Passive Attacks means hackers only interested in the data which have already presented in the database. In general, there are three ways to do this attack: Static leakage, linkage leakage and dynamic leakage [6].

3) Active Attacks
In active attacks, hackers not only get the information from the database, they also modify the actual data improperly. The cipher text value can be replaced with generated value, different cipher value and old version value.

4) SQL Injection Attack
The first three attacks are all based on the traditional computer environment, but with the development of Internet, more and more database server as a backend for web application. Under this situation, the attack aimed at web database began to appear, one of the major attacks is SQL injection attack. Hackers can let the server run the statement automatically. According to the Global Internet Report which published by Internet Society, the number of Internet users reached 3 billion at the end of 2016 [7]. So the data which stored in the web database is so huge and valuable unprecedented. Therefore, SQL Injection Attack is considered as the most dangerous type of attack on the up-to-date database.
3. SQL Injection
As we have discussed above, the SQL Injection is a new type of
attacks which aimed at database under web environment. Before
we continue our discussion about this attack, we'd like to sketch
how database works on the web application. The following is a
simple model about this:

![Figure 3.1 A Simple Model.](image)

As we can see from this figure, usually database management
system run on the top of an operating system and provide
the security associated with a database. But under web environment,
database no longer only exists in the local, so it becomes more
vulnerable. Hackers can access the database even those stored in
the local by web API very easily. And SQL injection is the most
common method hackers used.

3.1 Definition of SQL Injection Attack
According to the figure 3.1, we can know that SQLIA (SQL
Injection Attack) is an attack on web-application server. Because
of the web developers' neglect, the attackers can add the illegal SQL
code to the input box of a web form [8], this SQL query generated
based on user input, crafted with SQL keywords, operators, and
strings which execute in the database server [9]. Servers will run
this malicious SQL queries and by this way, attackers can access
the database and manipulate the database, like get the private
information, change the data even delete some useful data.

3.2 Evolution of SQL Injection Attack
Even though the concept of database and database security
appeared in early, the SQL Injection was first described just 20
years ago. The first SQL Injection incident was reported in 1996
and in 1998, Rain Forest Puply published an article in Phrack 54
which elaborated the idea of SQLIA firstly [10]. In February 1999,
Allaire published a warning "Multiple SQL Statement in Dynamic
Queries". At same year, RFP and Matthew Astley also gave a
warning that "NT ODBC Remote Compromise". Puppy published
second article about SQL Injection on 2000, which named "How I
hacked Packestorm - A look at hacking wthreads via SQL", in
this article, Puppy explained how SQL Injection works. In October
of the same year, Chip Andrews first used "SQL Injection " in
public, so far "SQL Injection" became a particular research field of
database security. January 2002, Chris Anley published a paper to
discuss this type of attack profoundly. This document discusses in
detail the common "SQL Injection" technique [11]. In 2004, at
Blackhat meeting, 0x90.org published their new tool used to
process SQL Injection, which was the first SQL Injection tool. Ten
years ago, SQL Injection evolved out of an entirely new attack type,
it can attack any dynamical ASP pages which have SQL Injection
vulnerabilities[12]. SQL Injection became more and more complex
dangerous, almost every year there are a large number of
website suffer from SQL Injection Attack.

3.3 SQL Injection Attack Process
Before we find out the method to detect and prevent SQL Injection,
we should firstly figure out its work principle. The figure 3.2 show
the basic process of SQL Injection:

![Figure 3.2 Process of SQL Injection Attack.](image)

According to above, we can get the three phases of a simple SQL
Injection Attack:
1) A hacker state a malicious HTTP request to the web application.
2) Create some SQL Queries.
3) Submit these statement to the back end database and access it.
The attackers can attack database easily by SQL Injection because
the web developers use the strings unreasonably or they believe
users' input overly. Like the sample we mentioned above, the SQL
statement which hacker used is always true because of the Boolean
tautology, hackers can access the web application and use the
database as an DBA without the right password [8].

3.4 Types of SQL Injection Attack
As we have mentioned in introduction, the SQL Injection Attack is
flexible and diversified, it has a lot of different type, we can classify
it by two ways, first is its goal and second is the SQL query it uses.

3.4.1 By goal
1) Getting Data: This type of attack is the most common because
the data which stored in the web application are always valuable,
like SSN, Bank Account, credit card password and other private
information.
2) Modifying Data: Sometime the hackers want to access the
database because they want to change the data, such as delete some
record or rewrite the original data. This type of attack is not very
common but the result of this attack is always disastrous.
3) Changing the schema of database: The goal of this attack not
only limited at records but the structures of databases, like the table
names, attributes names and data types of attributes.
4) Cutting down the connection between server and database:
This type of attack will make the database to web application
shutdown, thus denying service happen.
5) Controlling system remotely: This kind of injection is the most
dangerous because it allows hackers to gain control of the whole
system [13].

3.4.2 By type of SQL Query
1) Tautologies: This kind of attack injects SQL tokens to the
conditional query statement to be evaluated always true [14]. It
used to get the authentication control without know the password.
Because it uses some statements like (1=1), the query will be
always true.
2) Illegal/Logically Incorrect Queries: Hackers can use the error
messages to find vulnerable parameters in the web application, so
attackers will send the wrong queries on purpose. Like following example:

Figure 3.3 Sample of illegal incorrect query

3) Union Query: By using UNION, attackers can add the injected query to the safe query and then get the information from the application database.

4) Piggy-backed Queries: In this type of injection, attackers use query delimiter, such like ",", in general, this delimiter is used to commit the queries. By this way, attackers can add some illegitimate queries after a legitimate query.

5) Alternate Encodings: In this technique, attackers use different encoding method to inject database, such as ASCII, Unicode.

6) Inference: By this type of attack, intruders change the behavior of a database or application. There are two well known attack techniques that are based on inference: blind injection and timing attacks.

3.5 Perniciousness of SQL Injection Attack

The results of SQLIA can be disastrous because the attackers can get sensitive data from database and modify it even gain the permissions to control the whole system. The main perniciousness of SQLIA shows as following [15]:

Authorization: Critical and secret data can be manipulated by the attackers who do not have such privilege.

Authorization: Attackers can log in as a normal user without entering the right user name or password.

Confidentiality: If a database suffers from SQLIA, the private data will become transparent. Actually, this is the most serious perniciousness of SQLIA.

Integrity: By a successful SQLIA not only an attacker reads sensitive information, but also, it is possible to change or delete this private information.

4. Detection and Precaution of SQL Injection

As discussed earlier, SQL Injection Attack is the most dangerous attack on database. So detecting and preventing it as early as possible is pretty important for ensuring the safety of valuable information. Generally, the detection and prevention approaches for SQL Injection Attack can be categorized into post-generated and pre-generated approaches. In this part, we will first discuss these two approaches including the newest researches clearly. In addition, we will also summarize several newest researches about SQLIA detection and precaution method.

4.1 Pre-generated Approaches

Pre-generated, as the name suggests, programmers follow some guidelines for SQLIA detection during web-application development. This approach depends on the effectiveness of the development. This approach depends on the effectiveness of the tools applied to detect code that causes SQLIA [9]. The following is some pre-generated approaches:

4.1.1 Pixy

Pixy is a static analysis tool that used to detecting the vulnerabilities of web application, it was published by N Jovanovic and C Kruegel in 2006. This tool uses static source code analysis, which including flow-sensitive, interprocedural and context-sensitive dataflow analysis, alias and literal analysis [16]. But the drawback of this tool is it is open source so attackers can attack by finding the vulnerabilities in this tool.

4.1.2 Program Query Language

This tool combines the static analyzer with the dynamic analyzer. The static analyzer finds all potential matches conservatively using a context-sensitive, flow-insensitive, inclusion-based pointer alias analysis. Static results are also useful in reducing the number of instrumentation points for dynamic analysis. In the other hand, the dynamic analyzer instruments the source program to catch all violations precisely as the program runs and to optionally perform user-specified actions [17].

4.2 Post-generated Approaches

Post-generated detection approaches are useful for analysis of dynamic or runtime SQL query, generated with user input data by a web application, this method executes before posting a query to the database server [9]. It based on the initialization of trusted or untrusted strings, which are developer-dependent.

4.2.1 Positive tainting and Syntax aware evaluation

Positive tainting can address problems caused by incompleteness which may lead to false positives. According to Seltzer and Schroeder: in case of incompleteness, positive tainting fails in a way that maintains the security of the system [18]. The syntax-aware evaluation will be performed before the query strings are sent to the database and the keyword and operators will contain one or more characters without trust markings.

4.2.2 Context Sensitive String Evaluation

In this way, after user input the strings, the tool will first regard it as non-trusted data. Then by analyzing the data from the application which is regarded as trusted, the tool can get some characters of valid inputs and finally these characters will be used to determining the legitimacy of users' input and the unsafe characters will be removed [19].

4.2.3 Parse tree evaluation based on grammar

A parse tree is a data structure for the parsed representation of a statement. When hackers inject SQL into a database query, the parse tree of the intended SQL query and the resulting SQL query do not match [20].

4.3 Newest Research

The methods we have mentioned above are useful but in recent years, with the development of advanced algorithm and machine learning, many new detection methods have emerged. In this part, we will introduce three newest detection method, which are based on hash algorithm, and SVM respectively.
4.3.1 Using hash algorithm to detect SQL Injection

This method is raised by Maki Mahdi in 2016, to detect the SQLIA, the authors built a system which contains three models, the first one is web crawler, this model used to make a interaction between the detection system and the web application. The system can get the information from website including status-code, cookies and message-body and HTML content. Then, the system uses a injector model to send invalid payload to the affected parameter in the HTTP request. Finally, by using analyzer model, the system can use the hash value to detect if web has vulnerability [21].

4.3.2 SQLiGoT: Detecting SQLIA using SVM

In this paper, the authors built a novel model to detect SQL injection by modeling SQL queries as graph of tokens and using the centrality measure of nodes to train a Support Vector Machine (SVM) [22]. The core of SQLiGoT includes the following processes:

1) transforming the SQL query to a sequence of tokens preserving its structural composition.
2) generating a graph consisting of tokens as the nodes and connection between them as weighted edges.
3) Using the degree centrality measure of nodes to train the SVM classifier.
4) Using the SVM model to address the malicious queries at runtime.

By using the well-defined substitution scheme, we can transform the normal SQL query into the tokens graph easily. We wouldn't pay a lot of attention on this part, what we really care about is the SVM model which using to training and testing the SQL queries.

In this article, author uses three types of SVM models: single SVM system, Two SVM system and Three SVM system. Before elaborating these system, we should declare the variables which will be used in the systems.

4.3.2.1 Variables

1) Undirected graph: the weight between \( t_i \) and \( t_j \) is not related to their order of occurrence, that is, \( \omega_{ij} = \omega_{ji} \).

2) Directed graph: the weight between \( t_i \) and \( t_j \) is related to their order of occurrence, that is, \( \omega_{ij} \neq \omega_{ji} \).

3) \( s \) is the sliding window, that is, the number of tokens considered as related to each other within the window.

4) \( g \) is the number of tokens between two particular tokens.

5) Uniform weighting:
\[
\omega_{ij} = \begin{cases} 
1 & \text{if } 0 \leq g \leq s - 2 \\
0 & \text{otherwise}; 
\end{cases}
\]

6) Proportional weighting:
\[
\omega_{ij} = \begin{cases} 
s - g - 1 & \text{if } 0 \leq g \leq s - 2 \\
0 & \text{otherwise}; 
\end{cases}
\]

7) An adjacency matrix \( A \), and \( a_{ij} = \omega_{ij} \).

8) Degree centrality: \( C_D[i] = \deg(v_i) = a_{ii} + \sum_{j=1}^{n} a_{ij} \)

9) Indegree centrality: \( C_{\text{Di}}[i] = \deg^-(v_i) = \sum_{j=1}^{n} a_{ij} \)

9) Outdegree centrality: \( C_{\text{Do}}[i] = \deg^+(v_i) = \sum_{j=1}^{n} a_{ij} \)

4.3.2.2 Systems

1) Single SVM system

Using directed graphs, SVM 1 trained by indegree and SVM 2 trained by outdegree. If the sum of two output is +2, this query is regarded as injected. If the sum is -2, it is normal. But if the sum is 0, the query will be send to database but also need to send a warning to DBA.

2) Two SVM system

Using directed graphs, SVM 1 trained by indegree and SVM 2 trained by outdegree. If the sum of two output is +2, this query is regarded as injected. If the sum is -2, it is normal. But if the sum is 0, the query will be send to database but also need to send a warning to DBA.

3) Three SVM system

Using directed graphs, SVM 1 trained by degree, SVM 2 trained by indegree and SVM 3 trained by outdegree. If the sum is positive, it means that at least two of models regard the query as injected, so the query is rejected. If the sum is negative, the query will be regarded as normal.

According to the experimental result, even though the three-SVM-system has the second highest recall rate but it does not produce suspicious queries, so the author tests few real web applications using three-SVM system. The following is the result.

<table>
<thead>
<tr>
<th>App</th>
<th>Queries</th>
<th>Precision</th>
<th>Recall</th>
<th>FPR</th>
<th>Accuracy</th>
<th>F-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bookstore</td>
<td>68%</td>
<td>99.95</td>
<td>99.77</td>
<td>0.34</td>
<td>99.75</td>
<td>99.86</td>
</tr>
<tr>
<td>Forum</td>
<td>56%</td>
<td>99.93</td>
<td>99.77</td>
<td>0.35</td>
<td>99.75</td>
<td>99.85</td>
</tr>
<tr>
<td>Classifieds</td>
<td>61%</td>
<td>99.94</td>
<td>99.51</td>
<td>0.31</td>
<td>99.54</td>
<td>99.72</td>
</tr>
<tr>
<td>NewsPortal</td>
<td>61%</td>
<td>99.92</td>
<td>99.68</td>
<td>0.31</td>
<td>99.68</td>
<td>99.80</td>
</tr>
<tr>
<td>JobPortal</td>
<td>61%</td>
<td>99.96</td>
<td>99.69</td>
<td>0.30</td>
<td>99.69</td>
<td>99.83</td>
</tr>
</tbody>
</table>

5. REFERENCES


