# Research on SQL Injection and Defense Technology

Zongshi Chen, Mohan Li, Xiang Cui, and Yanbin Sun(✉)

Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, China
sunyanbin@gzhu.edu.cn

**Abstract.** With the rapid development of Internet technology, more and more dynamic web sites based on the B/S three-tier architecture have been established. At the same time, the security issues exposed by the websites are increasing, and the situation is not optimistic. Today, a large number of Web systems use a database to store various data of a website, which may be the user's personal information, or may be a company's trade secret information. If this information is leaked, it is a huge loss and risk to the individual or the company. SQL injection attacks can achieve the purpose of obtaining illegal data, so it is conceivable that the harm of SQL injection is huge. From the point of view of SQL injection, SQL injection attacks are still one of the most common and most dangerous attacks. This paper introduces the concept and technical principle of SQL injection attack, introduces the type of SQL injection, analyzes the basic implementation process of SQL injection attack, and finally gives the defense method of preventing SQL injection and summarizes some researches on SQL injection.

**Keywords:** SQL injection · Web security · Database security

## 1 Introduction

The security risk of injecting vulnerabilities is always in the OWASP top 10 [1] most serious security risks from 2004 to 2017. An injection vulnerability occurs when untrusted data is sent to the interpreter as part of a command or query. Injection vulnerabilities include SQL, NoSQL, OS, and LDAP injection. The focus of this article is on SQL vulnerabilities. SQL vulnerabilities are actually what we call SQL injection attacks. In 2000, Rain Forest Puppy released a report on SQL injection called "How I hacked PacketStorm" [2]. Since then, researchers have never stopped researching SQL injection. So far, research on SQL injection has done a lot of work.

SQL injection attacks have three characteristics. The first feature is that there are many variants, and different attackers have different SQL injection methods. The second feature is that the attack is simple. There are many SQL injection attack tools on the Internet. Those who know nothing about SQL injection only need to use these tools to easily attack or destroy the target website. The third feature is that the attack is extremely harmful. Due to the defects of the web language itself and the lack of


X. Sun et al. (Eds.): ICAIS 2019, LNCS 11635, pp. 191–201, 2019.
https://doi.org/10.1007/978-3-030-24268-8_18

developers with secure programming, most web application systems have the possibility of being attacked by SQL injection, and once the attacker succeeds, the attacker can Control the entire web application system to make any modifications or steals of the data, and the destructive power is reached to the extreme. The harm caused by SQL injection is even more serious than Distributed Denial of Service (DDOS) [3]. DDOS only makes the network paralyzed, but SQL injection will lead to more serious information leakage, network control and so on. Therefore, it is necessary to learn SQL injection.

The second section of this article will describe the concepts and principles of SQL injection. The third section describes the type of SQL injection. The fourth section briefly describes the general process of SQL injection attacks. The fifth section focuses on the defense methods of SQL injection and the methods of learning SQL injection against others.

## 2 The Concept and Principle of SQL Injection

### 2.1 The Concept of SQL Injection Attack

The definition of SQL injection attack is that the attacker uses the SQL security vulnerability existing in the website, inserts malicious SQL statements into the user input parameters, and finally spoofs the database server to execute malicious SQL commands to illegally obtain sensitive information such as user passwords.

The user input parameter is data input by the user in the webpage, for example, the user name and password entered in the login page belong to the user input parameter. Attacker can directly access the database and obtain data through a SQL injection attack, which poses a great security risk to the system.

### 2.2 The Technical Principle of SQL Injection

The principle of SQL injection is as follows. First, construct special inputs based on the SQL syntax, and then pass those inputs as parameters to the web application. After that, the maliciously constructed SQL command is injected into the website database through the WEB server for execution. Eventually perform various unauthorized operations on the database, such as query, update, delete, access, and so on.

The essence of SQL injection is the append command. The default SQL command of the website daemon is a normal command called “constant” in the program, and the user-controlled input such as user name, password, etc. is called “variable”. SQL injection is implemented by spelling together constants and variables as SQL instructions. For the convenience of understanding, the SQL injection flow is shown in Fig. 1.

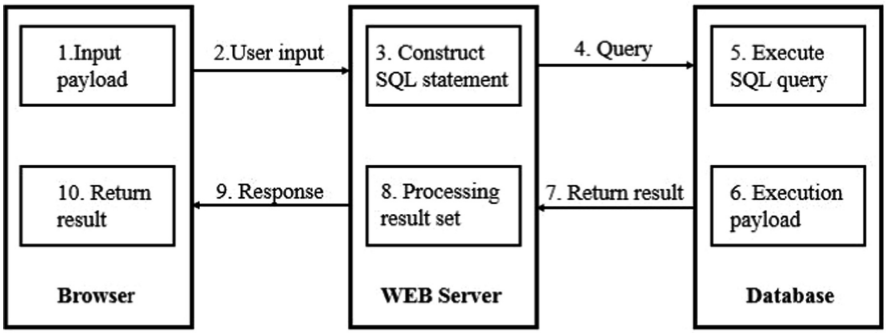


**Fig. 1.** SQL injection

The login verification module is taken as an example to illustrate the implementation principle of the SQL injection attack.

In the web application login verification program, there are two parameters "username" and "password". The program performs the authorization operation by the user's input username and password. The principle is to authorize access by looking up the results of the username and password in the user table.

Assume that the procedure for background login verification is as follows, taking the PHP code as an example [4]:

```
<?php
$username = $_POST['username'];
$password = $_POST['password'];
$sql = "Select * from users where username='$username' and password='$password'";
$result = mysql_query($sql);
if (!result) {
   die('<pre>' . mysql_error() .'</pre>' );
}
```

After the user enters the correct username and password (for example, the correct username: admin, password: password), the SQL query executed by the database is: select * from users where username = 'admin' and password = 'password'. As a result, there is a piece of data in which the user name is "admin" and the password is "password" in the query data, and finally enters the background successfully. However, when the user enters the username and password as Alice' or '1' = '1, the database query statement is: select * from users where username = 'Alice' or '1' = '1' and password = 'Alice' or '1' = '1'. Since the content input by the user is not detected, the database server is equivalent to executing the statement: select * from users, regardless of whether there is Alice' or '1' = '1 user in the database, the data will be returned and the login will be successful. Thus, the harm of SQL injection can be imagined.

## 3 Type of SQL Injection

The type of SQL injection can be divided into 5 types of SQL injection attacks according to the SQL syntax used in SQL injection, which are error injection, joint query injection, multi-state query injection, boolean-based blind injection and time-based blind injection [5]. Below I will describe these five SQL injection types one by one.

### 3.1 Error Injection

The main reason which causes the error injection is that the server does not close the error echo. When the attacker sends some maliciously constructed input query parameters to the website, some error information of website server can be obtained. According to the error information, some key information of the server is leaked.

The error message is the error reported by the database server executing the wrong SQL statement, not the error message designed by the web developer. For example, if we enter "select * from stu–+" at the SQL injection point, and "stu" is a non-existent database table that is randomly entered, it will return an error like this: Table 'dvwa.stu' doesn't exist. From this error message we got the name of the database.

### 3.2 Joint Query Injection

The joint query injection is to obtain the data of the database by connecting two different SQL statements through "UNION".

Specifically, the first SQL statement is generally provided to the user's normal query sql statement, and the second SQL statement is generally constructed by the attacker, they are connected through UNION. However, it should be noted that the number of fields in the query of two SQL statements in the UNION query must be the same, otherwise the SQL statement will not be successfully executed. For example, "select Firstname, Surname from users union select 1, 2 from users", the "Firstname" and "Surname" fields are provided in the SQL query provided by the website, and two fields must be constructed after constructing the SQL statement. We generally use "order by" to guess the number of fields in the original SQL statement.

### 3.3 Multi-statement Query Injection

Multi-statement query injection is also known as stacked query injection. In SQL statements, the semicolon ";" is used to indicate the end of a statement, and the next statement can be constructed after the end of an SQL statement. Compared with federated query injection, multi-statement query injection can construct any type of statement, and can perform database operations such as deletion, update, and addition.

For example, if you construct "1'; drop table user#" in the user input parameter, the database will actually execute: select * from users where id = '1'; drop table test#, the drop table operation will be directly executed after the query ends.

### 3.4 Boolean-Based Blind and Time-Based Blinds

Blind injection based on Boolean and blind injection based on time are theoretically blind injection. Blind injection means that the WEB server shields the error message generated by the SQL server returned to the user, and the attacker cannot directly obtain useful information through the error information of the database.

The difference between a Boolean blind and a time-based blind is that Boolean injection can be easily judged by using the customized page information returned by the web page. Whether the malicious input we construct is feasible, whether it can be injected, and time-based blindness. Note that when the page does not return any error information or does not output the data information detected by the joint query injection, the information acquisition can only be performed by using a function in the database that can extend the response time.

For example, when the time-based blind injection target is Mysql database, we can use the sleep() function in the if() function to perform SQL injection attacks. The following code is a SQL statement constructed for time blind injection, the function is to get the length of the current database name:

```
Select FirstName, Surname form users where id=1' and if (asci (substr (database( ), 1, 1 ) )
= 115, sleep(10), 1 ) --+
```

These kinds of SQL injection types are mainly aimed at constructing malicious SQL statements to obtain database data by using different SQL syntax used in SQL injection, and there is no difference in essence.

## 4 The General Process of SQL Injection Attacks

SQL injection attacks can be done manually, or through tool software such as Ming Xiaozi, D, sqlmap, etc. But no matter which kind of attack method is used, basically it is first to construct a special statement to detect the SQL vulnerabilities existing in the Web program, and then perform illegal operations according to the corresponding vulnerabilities, and finally obtain the database data and obtain the highest authority of the server.

We all know that the simplest and most basic SQL query statement is “select field name from table name”. It is not difficult to conclude that if we want to get the data information in the database, we must first know the table name and field name. At the same time, there may be slight differences in database functions for different databases, so in order to construct SQL statements effectively, we must first know which database the website uses.

The above knowledge is not difficult to get, SQL injection attacks generally go through the process:

1. Determine whether there is an injection point and whether it can be injected;
2. Guess which database, get the basic information of the database, such as version;
3. Guess the name of the table, get the name of the table;
4. Guess the field name and get the field name;
5. Guess the field value and get the field value.

## 5 Defense of SQL Injection Attacks

The harm caused by SQL injection attacks is terrible. The harm is not serious, it may only tamper with the information of the web page. If the harm is serious, the sensitive information of the database user or the private information will be leaked. Even the entire web server will be controlled by the hacker. This is very dangerous.

In order to minimize the loss caused by SQL injection attacks and increase the difficulty of SQL injection attacks, we can implement some preventive strategies and measures. We can generally prevent SQL injection attacks at the code and platform layers.

### 5.1 Defense at the Code Level

**Verify User Input**

User input verification is the process of testing the input received by the application to ensure it meets the criteria defined in the application. It can be as simple as limiting a parameter to a type, or as complex as using regular expressions or business logic to validate input [6]. The method of verifying user input can be simply divided into blacklist filtering and whitelist filtering.

Blacklist verification means rejecting all input that appears in the list. Usually, it is determined whether the input is accepted by determining whether there is a sensitive character. The sensitive characters are usually single quotes(’), double quotes(”), semicolons(;), and so on.

Whitelist verification means that only the specified conforming input is accepted. For example, the input data type, data content, data size [7], etc. are accepted by the user in the trusted list, otherwise it is not accepted.

Blacklist-based verification methods are much weaker than whitelist-based verification. It is unrealistic to rely on artificially adding all the sensitive characters to the blacklist. It is also possible that whitelisting may affect the use of normal users due to incomplete consideration. However, whether it is based on blacklist or whitelist defense methods, it has a great effect on preventing SQL injection.

Java, C#, HTML5, PHP, etc. all have their own format code to verify user input. Take PHP as an example, use the preg_math function to verify the parameters of the form.

```
<?php
$AdminName = $_POST[‘AdminName’];
If ( preg_math(“/^[A-Za-z] {12-16}$/D”, $ AdminName) ) {
   // Verification success!
}
?>
```

**Use Parameterized Query Statements**

One of the most fundamental reasons for the SQL injection attack is the ability to dynamically construct the generated SQL statement and then send it to the database for execution [8]. In order to prevent the characters entered by the user from being executed as SQL instructions, we can use parameterized query statements.

Parametric queries are currently the most effective defense against SQL injection attacks. The main idea of a parameterized query statement is as follows. First send the default SQL statement of the website to the database server for pre-compilation to generate a template, and then send the template back to the web server. After that, the user's input can only represent one parameter string, and can't be constructed as a new SQL statement. This eliminates the user-entered characters being executed as SQL statements, eliminating SQL injection.

Take PHP as an example to implement parameterized queries:

```
<?php
$AdminName = $_POST['AdminName'];
$AdminPw = $_POST['AdminPw'];
Result = Db.Execute("select admin where AdminName="'+ AdminName +"' and
AdminPw="'+ AdminPw +"'")
If(Result){
    // Verification success!
}
?>
```

**Use Hexadecimal Encoding**

Parametric queries can protect against all first-order SQL injection attacks, but they are powerless for second-order SQL injection attacks.

As mentioned above, SQL injection belongs to the first-order SQL injection, which means that the user input containing the attack payload is directly spliced into the dynamic SQL statement of the web application structure through the web input, so that the dynamic SQL statement is submitted to the database for execution and finally obtained unauthorized. Access technology [9], in which the attack load refers to the SQL statement that causes the original query logic of the SQL statement to be tampered with after being spliced by the SQL query, so as to implement the SQL statement that performs the illegal function [10].

Second-order SQL injection [11] is different from first-order SQL injection. The attack payload is injected into the WEB application and is not executed immediately to trigger the vulnerability. Instead, it is stored in the database and stored in the database when other operations are requested. The malicious code in the database will be retrieved and dynamically combined with the SQL statement of the WEB application into a new SQL statement to implement SQL injection attacks.

There are also many defense methods for second-order SQL injection, but they are not completely defense. In order to be able to almost completely prevent or even solve the second-order SQL injection attack, we can use ASCII hexadecimal encoding (that

is, use a 2-digit hexadecimal number to represent one character). The hexadecimal ASCII encoding of the string entered by the user prevents any executable SQL code from being inserted into the data, thus completely preventing the SQL injection attack [12].

All of the above mentioned are basic defense methods. There are also many people studying for more efficient defense methods or systems, and there are many achievements.

Muthuprasanna, et al. [13] propose a SQL detection and prevention technique, which is a fully automated technique that combines static analysis with runtime verification. In the static analysis phase, the author uses the Java String Analysis library for static analysis, representing SQL queries as finite state automata (SQL-FSM) and treating them as SQL graphs. In the runtime verification phase, it is illegal to use the static data structure to audit the SQL statement, so that it is safe to mark the currently detected statement. In addition to defending against SQL injection, the detection defense technology proposed by the author has two advantages. The first advantage is that there is no need to modify the code. The second advantage is that it speeds up web access.

Min and Kun proposed an idea of using regular expressions to detect and defend against SQL injection, and they also developed a simple system using Snort's regular expressions [14]. This system only needs two steps to detect and defend SQL injection. The first step is to design the rules of regular expressions. The second step is to execute regular expressions with snort. Regular Expressions Compared to SQL-FSM, regular expressions are more mature and easier to program and implement. However, this method also has disadvantages. The design of regular expression rules has a certain degree of limitation, which depends mainly on the performer's technology.

Encryption technology can not only do simple data encryption to ensure confidentiality, but also apply to SQL injection defense. An article mentions the use of Advanced Encryption Standard (AES) and RSA encryption for two-stage encryption to prevent SQL injection [15]. The author proposes to encrypt the user name and user password with AES, while the SQL statement is encrypted with the RSA algorithm. This defense method is feasible, but it also has shortcomings. The downside is the management and maintenance of the key, and this defense method is not useful for URL-based SQL injection. Mittal and Jena also use encryption to prevent and detect SQL injection. They proposed a technique based on Bitslice AES encryption to prevent and detect SQL injection [16]. Using this technique, you can prevent second-order SQL injection.

Naresh Duhan and Bharti Saneja proposed an efficient way to detect and prevent the notorious SQL injection problem [17]. Combine client-side validation with identity-based cryptography (IBC) to implement two layers of defense to solve SQL injection. The first layer of defense is client-side validation, which is implemented by applying static validation constraints to user input. The constraints can be the type of the data, the length of the string, or some special characters. The second layer of defense is IBC, which is the second line of defense against SQL injection. The key used for encryption is generated based on the combination of email addresses, and each user's key is unique. The advantage of this defense method is that it is efficient and easy to

implement. More importantly, it not only detects and prevents known types of SQL injection attacks, but also detects and prevents new types of SQL injection attacks.

Voitovych, et al. [18] made a defense system for SQL injection attacks. The main functional modules of this system are verification checkers, filters and error handlers. This system is very simple to implement, and it is not a new technology. It is essentially a method of checking and filtering to implement SQL injection defense.

Input validation is used to verify user input to ensure that the data entered is trustworthy and secure. Therefore, the way to input verification is to detect and defend against SQL injection, but it may have a problem of low detection accuracy. Lin, et al. [19] concluded that the input validation error occurred because only one filter rule was used. In order to solve the shortcomings of input verification, they proposed an automatic mechanism to improve the selection of filtering rules. They also made a defense system consisting of a test framework and a security gateway. The system they designed can achieve high detection rates.

### 5.2 Defense on the Platform Level

Defense against SQL injection attacks not only set protection mechanisms at the code level, but also protect against SQL injection attacks at the platform level. Insecure database configurations or vulnerabilities in the database platform can lead to the risk of SQL injection. So for SQL injection at the platform level, we can use the methods described below to defend against SQL injection.

**Correct Configuration of the Web Server**

The WEB server structure is large and complex, which makes the WEB server inevitably flawed in terms of security. Properly configuring the web server can reduce the risk of SQL injection. We can implement the security configuration of the WEB server from the following three aspects:

- Modify the server initial configuration: Once the server's brand is known by the hacker, the server's default configuration will be revealed, such as username and password. So we have to modify the initial configuration of the server.
- Install server security patches in a timely manner: By installing security patches, you can continuously improve your server to avoid vulnerabilities.
- Turn off the server's error message: According to the error information of the database, it is easier for the hacker to perform SQL injection, so we must configure the web server to block the error message.

**The Correct Configuration Database**

For the correct security configuration of the database, it is also one of the ways to reduce the risk of SQL injection. For example, database security configuration should follow the principle of least privilege [20]. Only grant the user the necessary permissions without over-authorization. The principle of least privilege can effectively reduce SQL injection and prevent important data leakage in the database. In addition to configuring the database, we mainly configure the database from the following two aspects:

- Modify the database initial configuration
- Upgrade the database in time

**Use Web Application Firewall (WAF)**
By configuring security parameters or rules such as WAF access control list, it can effectively prevent SQL injection attacks to a certain extent.

**Security Settings for Script Parsing**
For scripting languages such as ASP and PHP, some security settings are involved in their configuration files. We can increase the difficulty of SQL injection by properly configuring these security settings and reduce the risk of SQL injection.

In the case of PHP, we can set "magic_quotes_gpc" to "on" and "safe_mode" to "on".

## 6 Conclusion

With the advent of the information age, the number of Internet users has increased, and the number of websites has soared, and the web security situation has become increasingly severe. SQL injection attacks have become one of the most common attacks in web attacks. Therefore, learning and studying the principles of SQL injection attacks, the implementation process of attacks, and the knowledge of SQL injection defense have far-reaching significance for securing web security. In recent years, big data, cloud security [21], AI have also been rapidly developed, all of them are very useful researches for SQL injection.

**Acknowledgments.** This work is funded by the National Key Research and Development Plan (Grant No. 2018YFB0803504) and the National Natural Science Foundation of China (No. U1636215).

## References

1. Tan, J.: OWASP releases top ten web application security risks. Comput. Netw. (23), 52–53 (2017)
2. Puppy, F.R.: How I hacked PacketStorm: a look at hacking WWW threads by means of SQL —part 2. EDPACS **28**(3), 1–6 (2000)
3. Cheng, J., Xu, R., Tang, X., Sheng, V.S., Cai, C., et al.: An abnormal network flow feature sequence prediction approach for DDoS attacks detection in big data environment. Comput. Mater. Continua **55**(1), 095–119 (2018)
4. Ou, X., Yang, S.: Study on the principle and prevention technology of SQL injection attack. Digital Technol. Appl. (04), 216 (2016)
5. Halfond, W.G., Viegas, J., Orso, A.: A classification of SQL-injection attacks and countermeasures. In: IEEE International Symposium on Secure Software Engineering, vol. 1, pp. 13–15, March 2006
6. Shi, H., Ye, W.: SQL Injection Attack and Defense, 2nd edn. Tsinghua University Press, Beijing (2013)

7. Xu, J.: SQL injection attack principle and application in database security. Comput. Program. Skills Maint. (18), 104–106(2009)
8. Bo, Z.: Research on SQL injection attack and detection technology. Inf. Secur. Commun. Secur. (5), 90–92 (2010)
9. Herrero, Á., Corchado, E., Bajo, J., Pinzón, C.I., De Paz, J.F., Corchado, J.M.: idMAS-SQL: intrusion detection based on MAS to detect and block SQL injection through data mining. Inf. Sci. **231**, 15–31 (2013)
10. Kieyzuna, A., Guo, P.J., Jayaraman, K, et al.: Automatic creation of SQL injection and cross-site scripting attacks. In: Proceedings of the 31st International Conference on Software Engineering (ICSE), pp. 199–209. IEEE Computer Society, Washington, DC (2009)
11. Ollmann, G.: Second-order code injection attacks. Technical report. NGSSoftware Insight Security Research (2004)
12. Fu, X., Gong, X.: A general encoding method for solving SQL injection vulnerabilities. J. Yancheng Inst. Technol.: Nat. Sci. Ed. (1), 5–8(2015)
13. Muthuprasanna, M., Wei, K., Kothari, S.: Eliminating SQL injection attacks - a transparent defense mechanism. In: Eighth IEEE International Symposium on Web Site Evolution. IEEE Computer Society (2006)
14. Min, W., Kun, L.: An improved eliminating SQL injection attacks based regular expressions matching. In: International Conference on Control Engineering & Communication Technology. IEEE Computer Society (2012)
15. Balasundram, I., Ramaraj, E.: An Authentication scheme for Preventing SQL Injection Attack Using Hybrid Encryption (PSQL1-HBE) **53**(3), 359–368 (2011). ISSN 1450-216 X
16. Mittal, P., Jena, S.K.: A fast and secure way to prevent SQL injection attacks. In: Information & Communication Technologies. IEEE (2013)
17. Duhan, N., Saneja, B.: A two tier defense against SQL injection. In: International Conference on Signal Propagation & Computer Technology. IEEE (2014)
18. Voitovych, O.P., Yuvkovetskyi, O.S., Kupershtein, L.M.: SQL injection prevention system. In: Radio Electronics & Info Communications. IEEE (2016)
19. Lin, J.C., Chen, J.M., Liu, C.H.: An automatic mechanism for sanitizing malicious injection. In: International Conference for Young Computer Scientists. IEEE (2008)
20. Qi, C.: Web security development: SQL injection attacks and web page hanging horses. Programmer (7), 102–104 (2008)
21. Zhang, H., Yi, Y., Wang, J., Cao, N., Duan, Q., et al.: Network security situation awareness framework based on threat intelligence. Comput. Mater. Continua **56**(3), 381–399 (2018)