# SQL Injection and Prevention

*Network Security Class: Course Project

Li Kaixin
*58119132*
*Southeast University*
Nanjing, China
1321604615@qq.com

Jiang Zhuoyang
*58119125*
*Southeast University*
Nanjing, China
1050251573@qq.com

***Abstract*—A database system includes the database itself and database applications. MySQL can be used to manage the database and realize the functions of database applications. SQL injection is a network attack on the database itself against the vulnerability of the database application program. Through the interactive interface of the database application program, malicious code can be injected into the database management code or the database itself to achieve the attack target. In this experimental project, we will build a network attack and defense scenario about SQL injection, build the database and the application website corresponding to the database, analyze the SQL injection related vulnerabilities in the website, use different methods to achieve different kinds of SQL injection attacks and achieve different attack goals. At the same time, according to our analysis of SQL injection related vulnerabilities, we also implemented preventive measures for different kinds of SQL injection, which effectively prevented SQL injection attacks.**

## I. Introduction

SQL is a structured query language for operating database data. SQL will be used when the application data of web pages interact with the data in the background database. SQL injection is to modify and splice the input parameters of the original URL, form field or data package of the web page into SQL statements, which are passed to the web server, and then to the database server to execute the database command.

We need to understand the basic knowledge related to SQL and analyze the preconditions of SQL injection, so as to explain the SQL injection attack and defense in our project practice

### *A. Concepts of database, SQL language and MySQL*

Database is a very important technology in the field of computer application. It is an important branch of software technology. The database system consists of database application program, database management program and database itself.

SQL is a database query language and programming language. It is a relational database language, which is mainly used to manage the data in the database.

MySQL is a relational database management system. Relational database saves data in different tables instead of putting all data in a large warehouse, which increases speed and flexibility.

MySQL uses SQL language to realize the management function. Due to its small size, fast speed and low overall cost of ownership, especially the characteristics of open source, MySQL is generally selected as the website database for the development of small, medium-sized and large websites. We also choose MySQL as the database application management tool.

### *B. Interaction between database and Web App*

As we know, the database system consists of database itself and the database application program. So that, before the basic SQL injection attack, we need to analyze how web database applications interact with database .

After obtaining the key value entered by the user, the web page constructs the query string of SQL language with the obtained key value, and then sends the query string to the database to execute the SQL command. That is, the data entered in the form will eventually become part of the SQL command string executed by the database. Even if the user does not directly interact with the database at the database end, there is indeed a channel between the user and the database at the web page input end.

The implicit channel between the user and the database creates attack conditions for the database. If the writing method of the database interface is not properly protected, the user may attack the database itself by using the string acquisition in the code on the basis of analyzing and understanding the code of the database interface.

### *C. Attack and defense of SQL injection*

SQL injection means that the web application does not judge or filter the legitimacy of the user input data. The attacker can add additional SQL statements at the end of the query statements defined in advance in the web application, and realize illegal operations without the knowledge of the administrator, so as to deceive the database server to execute unauthorized arbitrary queries, so as to further obtain the corresponding data information. [1]

We will build a network attack and defense scenario about SQL injection, build the database and the application website corresponding to the database, analyze the SQL injection related vulnerabilities in the website [2], and use different

methods to realize different kinds of SQL injection attacks and achieve different attack targets. At the same time, according to our analysis of SQL injection related vulnerabilities, we also implemented preventive measures for different kinds of SQL injection, which effectively prevented SQL injection attacks.

## II. Experimental Environment Construction

Before starting the attack and prevention, we need to build an attack scenario. Suppose there is a database for storing the scores of Southeast University Students' network security final exam. Students can query their final exam scores through a query Web page built by the teacher.

Specifically, the database needs to store students' test scores and relevant identity information, while the query web page needs us to realize the login function, user management function (mainly password modification) and score information display function.

Therefore, we introduce the experimental process of attack scenario construction from three steps: database construction, query website construction and query website connection database.

### A. Database construction

We need to build a database to store students' test scores. In addition to storing students' test scores, the database also needs to store the corresponding students' name, student number, login password and QQ to identify students and match students' information with their test scores.

At the same time, the student number can be used as the user name, together with the login password, for the login and user management of the score query web page.

The construction of database is mainly realized by the following steps.

*1) Login the database:* Before all operations, we need to login the data base as below.

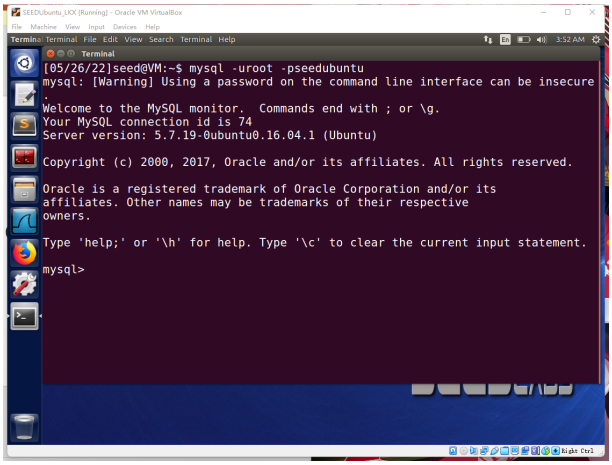


Fig. 1. Login the database

*2) Create the database:* To create a database, We use command **create database studentGrade;** to create a database named studentGrade.

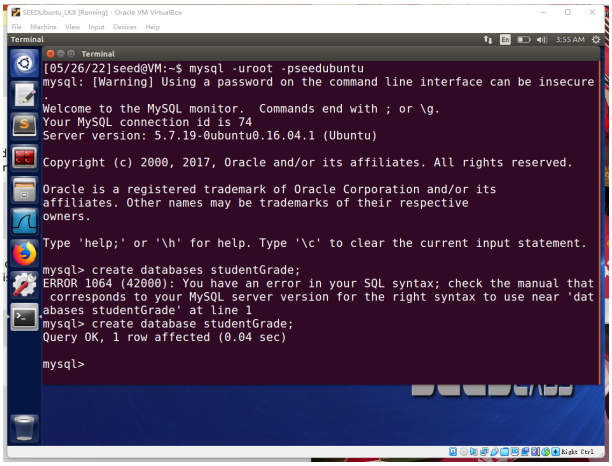


Fig. 2. Create a database

*3) Access the database and create a table:* Then, We use command **use studentGrade;** to access the studentGrade database and **create table finalGrade (...);** to create a table named finalGrade.

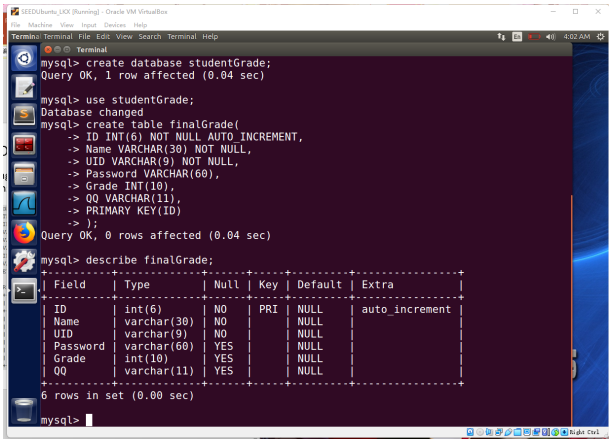


Fig. 3. Access the database and create a table

*4) Insert data into the table:* Finally we use command **insert into finalGrade (...) values (...), ...;** to insert data into the table.

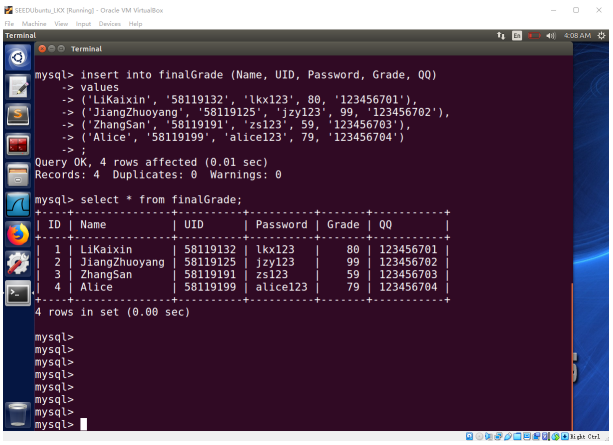


Fig. 4. Insert data into the table

## B. Enquiry website construction

After building the database, we need to build a supporting score query website, which is composed of three pages: user login page, user score information page and user management page.

First of all, the website needs a basic user login page, which allows students to enter their student number and login password to log in.

After the student enters the student number and login password to log in, you can jump to the score information page corresponding to the student's student number. The content of this page includes not only the student's test score, but also the student's relevant information, such as the student's name, student number and QQ.

Finally, the website needs a user management page to provide students with an interface to modify their login password.

Before the concrete construction, we need to analyze the relationship between the information of the database and the query web page. There are three main links:

- Connect the user login function of the website with the student number and login password in the database: the user login function requires to compare the entered student number and login password with the data in the database to determine whether the login is successful.
- Connect the user's score information of the website with the student's score and student related information in the database: the website needs to retrieve the name, student number, examination score and QQ corresponding to the login from the database and display them in the student's score information page after login.
- Connect the password modification function of the website with the login password in the database: the function of modifying the password on the user management page should be realized. When the operator modifies the password on the web page, the login password of the operator needs to be modified synchronously in the database

*1) Build a home page for query:* First, we build a simple home page for query results.

```
<html>
<head>
    <meta http-equiv="content-type" content="text/html;charset=utr-8;">
    <title>finalGrade</title>
</head>

<body>
    <h2> Check you final grade ! </h2>
    <form action="getdata.php" method="get">
    StudentID:<input type="text" name="UID"><br>
    Password: <input type="text" name="Password"><br>
              <input type="submit" value="Check">
    </form>
    <a href="changepassword.html">changeYourPassword</a>
</body>

</html>
```

Fig. 5. Build a home page for query

After the setup is successful, the UI effect of the login interface is shown as follows:

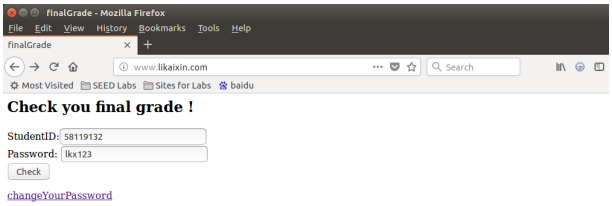


Fig. 6. UI effect of the login interface

*2) Database connection and query:* Then we use PHP language for database connection and query operations, here, we take the student ID and password as the query conditions, query results for the student ID, student name, student achievement and their QQ. This can complete the construction of user achievement information.

```
<html>
<head>
    <meta http-equiv="content-type" content="text/html;charset=utr-8;">
    <title>checkGrade</title>
</head>
<body>
<?php
    session_start();
    $uid = $_GET['UID'];
    $pwd = $_GET['Password'];

    $conn = new mysqli("localhost","root","seedubuntu","studentGrade");
    $sql = "SELECT UID, Name, Grade, QQ
            FROM finalGrade
            WHERE UID='$uid' and Password='$pwd'";

    $result = $conn->query($sql);
    if ($result) {
       // Print out the result
       while ($row = $result->fetch_assoc()) {
          printf ("StudentID: %s -- Name: %s -- Grade: %s -- QQ: %s\n",
                  $row["UID"], $row["Name"], $row["Grade"], $row["QQ"]);

       }
       $result->free();
    }
    $conn->close();

?>
</body>
</html>
                                                    1,1           All
```

Fig. 7. Database connection and query operations

After the user logs in successfully, the result information page is displayed as follows:


Fig. 8. Interface of query result

*3) Build a web page for user management:* Build a simple web page for user management (mainly to modify the password), which can be redirected to the main page of query results.

```
Terminal
<html>
<head>
    <meta http-equiv="content-type" content="text/html;charset=utr-8;">
    <title>finalGrade</title>
</head>
<body>
    <h2> Change your password! </h2>
    <form action="changepassword.php" method="post">
    StudentID:<input type="text" name="UID"><br>
    Old Password:<input type="text" name="oldpassword"><br>
    New Password:<input type="text" name="newpassword"><br>
                <input type="submit" value="Change">
    </form>
</body>
</html>
~
~
```

Fig. 9. Build a web page for user management

After the successful construction, the UI effect of the user management interface is shown as follows:'

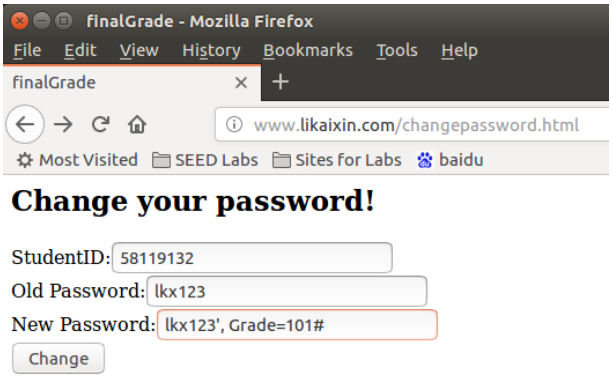


Fig. 10. Build a web page for user management

*4) Database connection and modification:* Here, we also use PHP language to connect and modify the database. We take the student ID and the old password as the query condition, and modify the information to the new password of the student. This can complete the modification of student information.

```
Terminal
<html>
<head>
    <meta http-equiv="content-type" content="text/html;charset=utr-8;">
    <title>finalGrade</title>
</head>
<body>
    <h2> Change your password! </h2>
    <form action="changepassword.php" method="post">
    StudentID:<input type="text" name="UID"><br>
    Old Password:<input type="text" name="oldpassword"><br>
    New Password:<input type="text" name="newpassword"><br>
                <input type="submit" value="Change">
    </form>
</body>
</html>
~
~
```

Fig. 11. Database connection and modification operations

## III. SQL Injection Attack

In the built attack scenario, we conduct the experiment of SQL injection attack. According to the existing scenarios, there are three kinds of attacks that can be carried out. The first is basic SQL injection attacks, the second is SQL injection attacks using curl, and the last is database modify. [3]

These three attacks can achieve different targets by using different vulnerabilities in SQL language to establish a database:

- We can directly log in to other people's score query page without password to view other people's scores
- We can directly use the URL to obtain the information of the complete student achievement database
- In the page of changing the password, we use the SQL writing vulnerability that the data and code are not separated to change our grades in the database and make our grades higher.
- In the page of modifying the password, we use the SQL writing vulnerability that the data and code are not separated to maliciously change the scores of others in the database and lower the scores of others.

### A. A Basic SQL Injection Attacks

Web application developers have developed a data input interface, which is intended to allow users to input some data in the interface to realize interactive functions.

Originally, users should only input data in the data input interface of the web page, but these input data will become a part of the SQL command in the code. Therefore, users can inject special code fragments disguised as data into the input interface, which contain special characters, such as "" character (used to comment the code), so that we can use these special characters to change the meaning of SQL sentences in the code to achieve the attack target.

We will use this vulnerability to inject code fragments disguised as data into the student ID input interface of the user's login page, so as to achieve the attack target of "logging in to other people's score query page without password and viewing other people's scores". The specific implementation is as follows.

For example, if we want to query someone's score without knowing his password but only know his ID, we can use the following input:

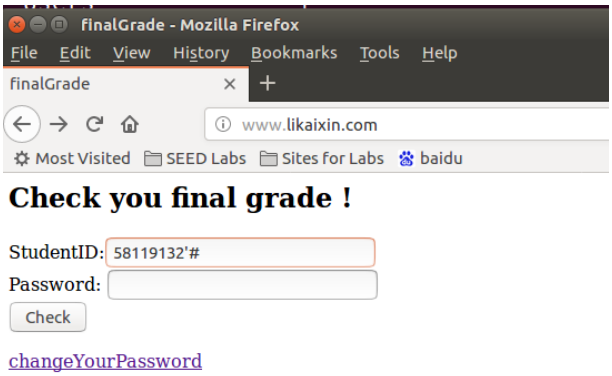


Fig. 12. Web SQL injection

After entering this information, the query statement is **select UID, Name, Grade, QQ from finalGrade where UID='58119132' #' and Password='';**:, which is equivalent to **select UID, Name, Grade, QQ from finalGrade where UID='58119132'**. So we can query the information we want. As shown in the figure below:

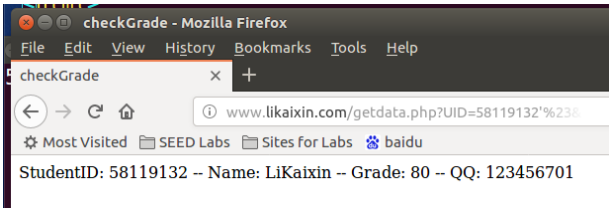


Fig. 13. Web SQL injection result

## B. SQL Injection Attacks using cURL

In addition to injecting code directly into the interactive interface of web pages, it is more convenient to launch attacks with command-line tools. In this way, we can easily automate attacks without entering the graphical user interface. Using the curl command, you can send malicious code disguised as data in the command-line tool.

We will use the curl command, write the URL in the command-line tool, and inject malicious code into the data interaction interface in the URL. The specific implementation is as follows.

For example, we could send the form directly from the following command line, but we would find that this command does not work.


Fig. 14. cURL SQL injection failed

This is because in HTTP requests, special characters in the attached data need to be encoded or they will be misinterpreted. Where space is encoded as %20, # as %23, and single quotes as %27. The following figure shows the curl command and query result obtained after encoding special characters:

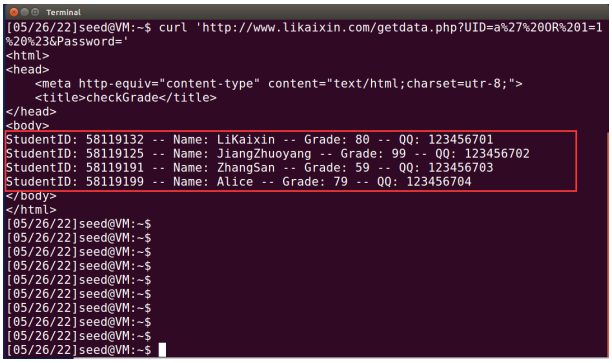


Fig. 15. cURL SQL injection succeed

## C. Database Modify

If there is a function interface for modifying data in the data interface of the web page, this function requires the user to input data in the web page, and then update the data in the database with the data entered by the user. In this kind of interface that will insert or change the data in the database, we will have the opportunity to change the database at the interface of the web page.

In our attack scenario, we can modify the student's password in the user management page of the website. The password modifying interface is an interface that will update the database data. We can use this interface to achieve the attack target of changing our own or others' grades. The specific implementation is as follows.

*1) Modify your own grade higher:* Let's assume that I (UID 58119132) am not satisfied with my score (80), and I want to change my score higher by SQL injection. Then I will type my UID and my old password, and type the new password as follows:

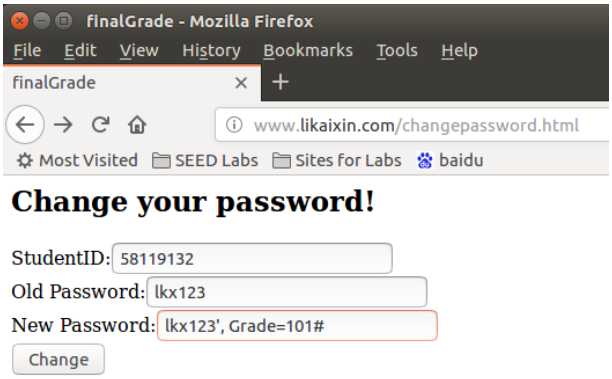


Fig. 16. Modify your own grade higher

After this information is typed, the query statement is:**update finalGrade set Password='lkx123', Grade=101# ' where UID='58119132' and Password='lkx123'**, which is equivalent to:**update finalGrade set Password='lkx123' where UID='58119132' and Password='lkx123', Grade=101**. So we can modify our score in the database.

```
mysql> select * from finalGrade;
+----+--------------+----------+----------+-------+-----------+
| ID | Name         | UID      | Password | Grade | QQ        |
+----+--------------+----------+----------+-------+-----------+
|  1 | LiKaixin     | 58119132 | lkx123   |    80 | 123456701 |
|  2 | JiangZhuoyang| 58119125 | jzy123   |    99 | 123456702 |
|  3 | ZhangSan     | 58119191 | zs123    |    59 | 123456703 |
|  4 | Alice        | 58119199 | alice123 |    79 | 123456704 |
+----+--------------+----------+----------+-------+-----------+
4 rows in set (0.00 sec)

mysql> select * from finalGrade;
+----+--------------+----------+----------+-------+-----------+
| ID | Name         | UID      | Password | Grade | QQ        |
+----+--------------+----------+----------+-------+-----------+
|  1 | LiKaixin     | 58119132 | lkx123   |   101 | 123456701 |
|  2 | JiangZhuoyang| 58119125 | jzy123   |    99 | 123456702 |
|  3 | ZhangSan     | 58119191 | zs123    |    59 | 123456703 |
|  4 | Alice        | 58119199 | alice123 |    79 | 123456704 |
+----+--------------+----------+----------+-------+-----------+
4 rows in set (0.00 sec)
```

Fig. 17. Modify your own grade higher result

*2) Maliciously modify others grade:* If I don't like ZhangSan and I want to modify his score to 0, but I only know his ID (58119191) and don't know his password, then I can maliciously modify his score in the following ways:

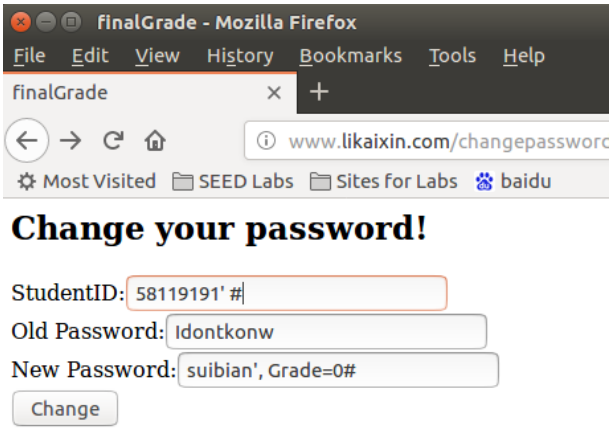


Fig. 18. Modify others grade

The principle is the same as above, and the modification results are as follows:

```
mysql> select * from finalGrade;
+----+--------------+----------+----------+-------+-----------+
| ID | Name         | UID      | Password | Grade | QQ        |
+----+--------------+----------+----------+-------+-----------+
|  1 | LiKaixin     | 58119132 | lkx123   |    80 | 123456701 |
|  2 | JiangZhuoyang| 58119125 | jzy123   |    99 | 123456702 |
|  3 | ZhangSan     | 58119191 | zs123    |    59 | 123456703 |
|  4 | Alice        | 58119199 | alice123 |    79 | 123456704 |
+----+--------------+----------+----------+-------+-----------+
4 rows in set (0.00 sec)

mysql> select * from finalGrade;
+----+--------------+----------+----------+-------+-----------+
| ID | Name         | UID      | Password | Grade | QQ        |
+----+--------------+----------+----------+-------+-----------+
|  1 | LiKaixin     | 58119132 | lkx123   |    80 | 123456701 |
|  2 | JiangZhuoyang| 58119125 | jzy123   |    99 | 123456702 |
|  3 | ZhangSan     | 58119191 | suibian  |     0 | 123456703 |
|  4 | Alice        | 58119199 | alice123 |    79 | 123456704 |
+----+--------------+----------+----------+-------+-----------+
4 rows in set (0.00 sec)
```

Fig. 19. Modify others grade result

## IV. SQL Injection Prevention

For the vulnerabilities attacked by SQL injection, we have mainly implemented two prevention methods, one is based on filtering, and the other is based on parameter value passing. [1]The former will filter out the characters that may produce malicious code, but it does not fundamentally solve the problem. The code and data are not separated, while the latter realizes the separation between code and data, which well prevents the occurrence of SQL injection.

### A. Filtering and Encoding Data

During SQL injection attack, the attacker writes SQL injection statements through various special characters. Therefore, to prevent SQL injection, it is necessary to standardize user input and ensure the security of data input. When specifically checking the variables input or submitted in the web interface, it is necessary to convert or filter the special characters that will affect the function of SQL statement, so as to effectively prevent SQL injection.

In our experimental scenario, before mixing the student number and other data provided by the user with the SQL

query code, first check the input data and filter out any characters that may be interpreted as code, such as "". Encoding special characters will tell the parser to treat the encoded characters as data rather than code. The specific implementation is as follows.

*1) Write prevention code with filtering function:* PHP's mysqli extension has a built-in method called **mysqli::real_escape_string()**. It can be used to encode the characters that have special meanings in SQL. The API is used in code as below:

```
    <title>checkGrade</title>
</head>
<body>
<?php
    session_start();
    $getuid = $_GET['UID'];
    $getpwd = $_GET['Password'];
    printf ("Before encoding: UID=%s, Password=%s\n<br>",
            $getuid, $getpwd);
    $conn = new mysqli("localhost","root","seedubuntu","studentGrade");
    $uid = mysqli_real_escape_string($conn,$_GET['UID']);
    $pwd = mysqli_real_escape_string($conn,$_GET['Password']);

    printf ("After encoding: UID=%s, Password=%s\n<br>",
            $uid, $pwd);

    $sql = "SELECT UID, Name, Grade, QQ
            FROM finalGrade
            WHERE UID='$uid' and Password='$pwd'";

    $result = $conn->query($sql);
    if ($result) {
       // Print out the result
       while ($row = $result->fetch_assoc()) {
          printf ("StudentID: %s -- Name: %s -- Grade: %s -- QQ: %s\n",
                  $row["UID"], $row["Name"], $row["Grade"], $row["QQ"]);

       }
       $result->free();
    }
    $conn->close();

?>
<br><br>
<b>PS: [SQL injection defend -- Filtering and Encoding Data]</b>

</body>
</html>
```

Fig. 20. Filtering and Encoding Code

*2) Write prevention code with filtering function:* Use the code with filtering function to realize the information interaction between web application and database.

For example, we first input a group of student id and password to query normally.

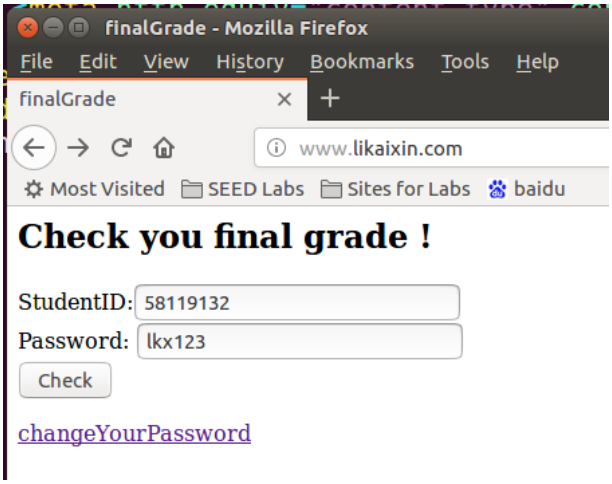


Fig. 21. Check your grade normally

As you can see, we get the normal result which has a normal encoding string without special characters as shown in the figure below:

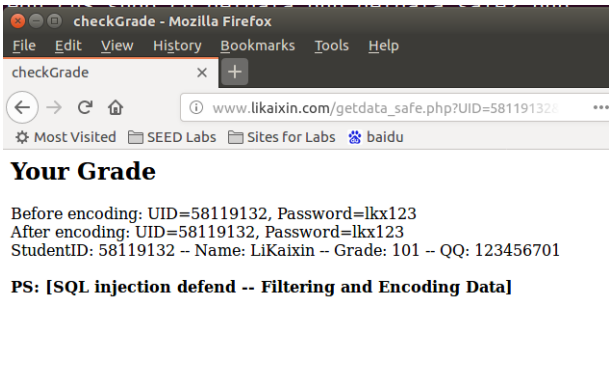


Fig. 22. Check your grade normally result

If we use SQL injection to enter information as below:

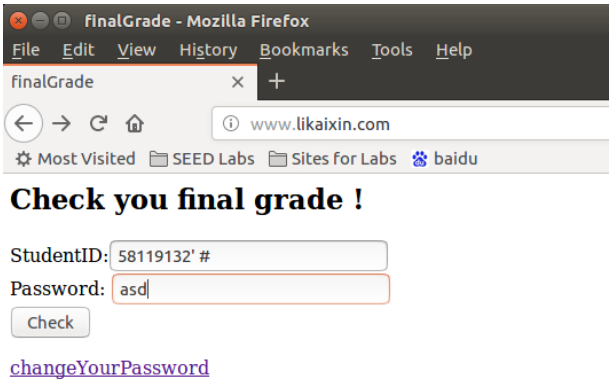


Fig. 23. Check your grade with SQL injection method

It can be seen that we cannot find the information we want, the special character will be filtered and the query result is shown in the figure below:

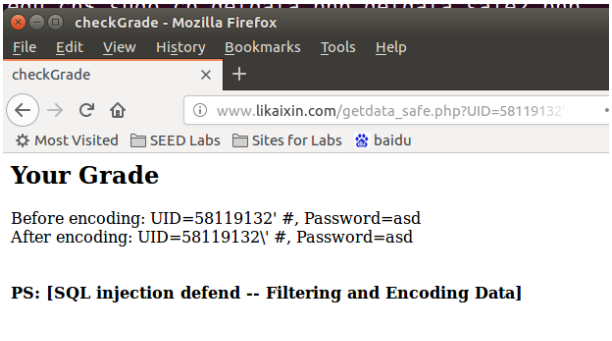


Fig. 24. Check your grade with SQL injection method result

## B. Prepared Statement

Because filtering or escaping methods do not address the root cause of the problem, data and code are still mixed together. So we introduced the following method.

When programmers write SQL language, it is forbidden to write variables directly into SQL statements. They must pass related variables by setting corresponding parameters. This inhibits SQL injection. Data input cannot be embedded directly into a query statement. At the same time to filter the input content, filter out unsafe input data. Alternatively, input variables can be passed as parameter values to maximize the defense against SQL injection attacks.

*1) Write prevention code with Prepared Statement:* We used PHP to write Prepared Statement defense code against SQL injection. It separates the data from the SQL Statement, filters the data, and then merges the data with the code.

```
<html>
<head>
    <meta http-equiv="content-type" content="text/html;charset=utr-8;">
    <title>checkGrade</title>
</head>
<body>
<h2>Your Grade </h2>
<?php
    session_start();
    $uid = $_GET['UID'];
    $pwd = $_GET['Password'];
    $conn = new mysqli("localhost","root","seedubuntu","studentGrade");
    if ($conn->connect_errno){
        echo "Failed to connect to MySQL: ".$conn->connect_error;
        exit();
    }

    $sql = "SELECT UID, Name, Grade, QQ
            FROM finalGrade
            WHERE UID=? and Password=?";

    $stmt = $conn->stmt_init();
    if ($stmt->prepare($sql)) {
        $stmt->bind_param("ss", $uid, $pwd);
        $stmt->execute();
        $stmt->bind_result($sid, $name, $grade, $qq);

        while ($stmt->fetch()) {
            printf ("StudentID: %s -- Name: %s -- Grade: %s -- QQ: %s\n",
                    $sid, $name, $grade, $qq);
        }
    }
    $stmt->close();
    $conn->close();
?>
<br><br>
<b>PS: [SQL injection defend -- Prepared Statement: Separation of code and data]</b>
</body>
```

Fig. 25. Prepared Statement with PHP

*2) Example:* For example, we first input a group of student id and password to query normally.

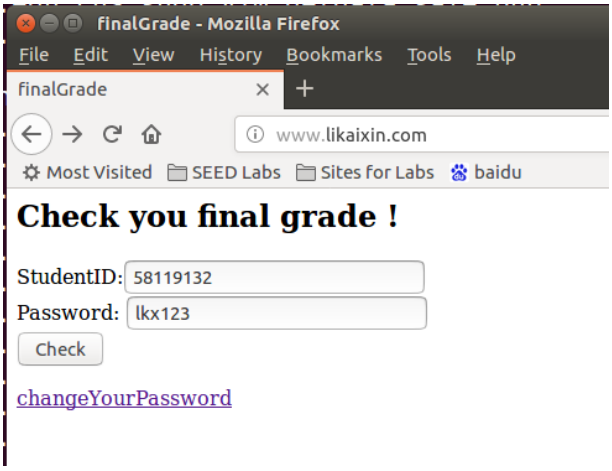


Fig. 26. Check your grade normally

As you can see, we get the correct result, as shown in the figure below:


Fig. 27. Check your grade normally result

If we use SQL injection to enter information,

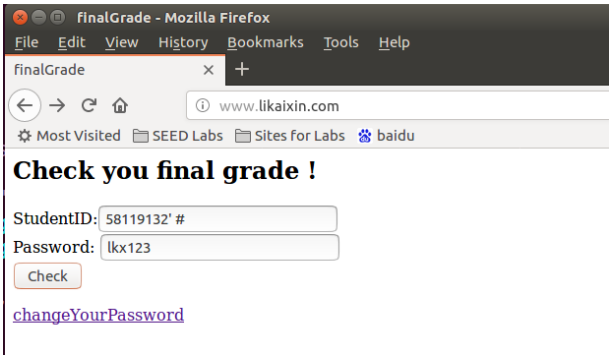


Fig. 28. Check your grade with SQL injection method

It can be seen that we cannot find the information we want, and the query result is shown in the figure below:

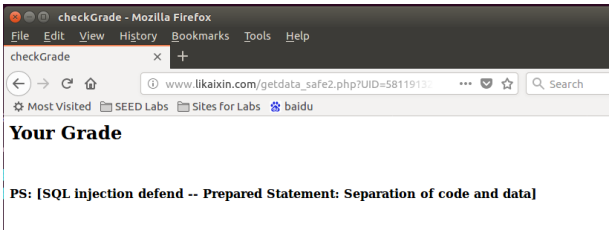


Fig. 29. Check your grade with SQL injection method result

## V. Summary

In this project, we have a brief tutorial of SQL, MySQL and Database system. We built a network attack and defense scenario about SQL injection which consists the database and the application website corresponding to the database.

We also learned the concept of SQL Injection attack and how to launch this type of attacks. Most important of all, we get a good command of the fundament cause of the vulnerability and how to defend against SQL Injection attacks.

## References

[1] Boyd S W, Keromytis A D. SQLrand: Preventing SQL injection attacks[C]//International Conference on Applied Cryptography and Network Security. Springer, Berlin, Heidelberg, 2004: 292-302.

[2] Halfond W G, Viegas J, Orso A. A classification of SQL-injection attacks and countermeasures[C]//Proceedings of the IEEE international symposium on secure software engineering. IEEE, 2006, 1: 13-15.

[3] Kindy D A, Pathan A S K. A survey on SQL injection: Vulnerabilities, attacks, and prevention techniques[C]//2011 IEEE 15th international symposium on consumer electronics (ISCE). IEEE, 2011: 468-471.
[4] Johari R, Sharma P. A survey on web application vulnerabilities (SQLIA, XSS) exploitation and security engine for SQL injection[C]//2012 international conference on communication systems and network technologies. IEEE, 2012: 453-458.