php代码安全特性——张晏龄

# 目录

[一、精度绕过缺陷 1](#_Toc5006)

[二、类型转换的缺陷 2](#_Toc23654)

[三、松散比较符的缺陷 4](#_Toc10837)

[四、sha1() md5()加密函数漏洞缺陷 11](#_Toc17091)

[五、字符串处理函数漏洞缺陷 14](#_Toc26635)

[六、变量覆盖漏洞 16](#_Toc4079)

[七．in\_array()强转类型 19](#_Toc326)

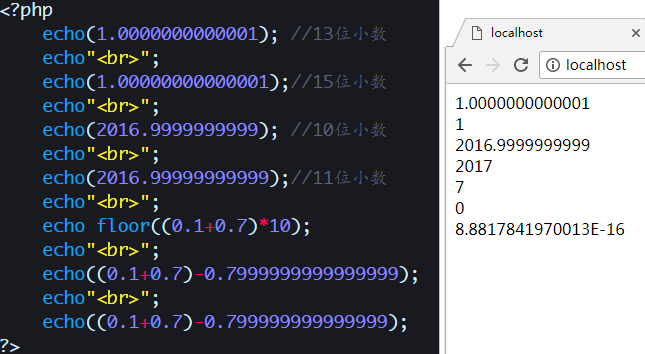
[参考文献： 19](#_Toc10108)

## 

## 一、精度绕过缺陷

### 理论

在用PHP进行浮点数的运算中,经常会出现一些和预期结果不一样的值，这是由于浮点数的精度有限。尽管取决于系统，PHP 通常使用 IEEE 754 双精度格式，则由于取整而导致的最大相对误差为 1.11e-16。非基本数学运算可能会给出更大误差，并且要考虑到进行复合运算时的误差传递。



以十进制能够精确表示的有理数如 0.1 或 0.7，无论有多少尾数都不能被内部所使用的二进制精确表示，因此不能在不丢失一点点精度的情况下转换为二进制的格式。这就会造成混乱的结果：

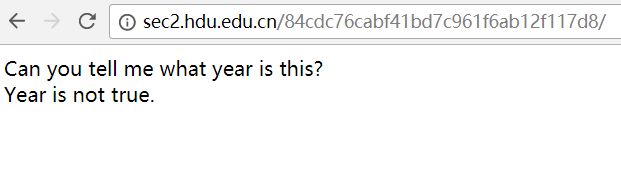
floor((0.1+0.7)\*10) 通常会返回 7 而不是预期中的 8，因为该结果内部的表示其实是类似 7.9999999999999991118…。

### 实践：问鼎杯2017

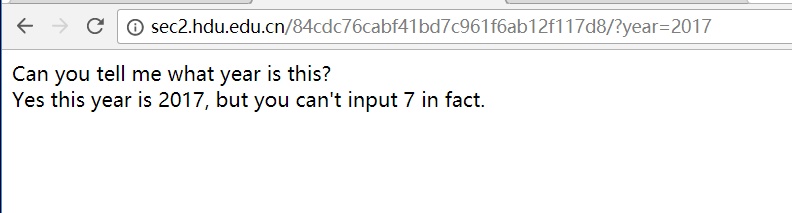
**考察点**

* PHP浮点精确度

### **write-up**



题目提及YEAR，即将2017值传给服务器 ?year=2017



提示：2017中不可出现7，利用PHP精度来绕过: ?year=2016.99999999999



## 二、类型转换的缺陷

### 理论

PHP提供了is\_numeric函数，用来变量判断是否为数字。PHP弱类型语言的一个特性，当一个整型和一个其他类型行比较的时候，会先把其他类型intval数字化再比。

## **实践：**is\_numeric()用于判断是否是数字，通常配合数值判断。

### **考察点**

* PHP类型转换缺陷

### **write-up**

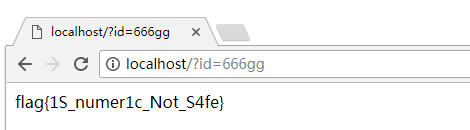


## 分析代码: id通过is\_numeric函数来判断是否为数字：

## 如果为数字的话，GG。

## （2）如果不是数字的话，和665进行比较，id的值大于665的时候输出flag。 PHP弱类型语言的一个特性，当一个整形和一个其他类型行比较的时候，会先把其他类型intval数字化再比

绕过：http://localhost/?id=666gg

、

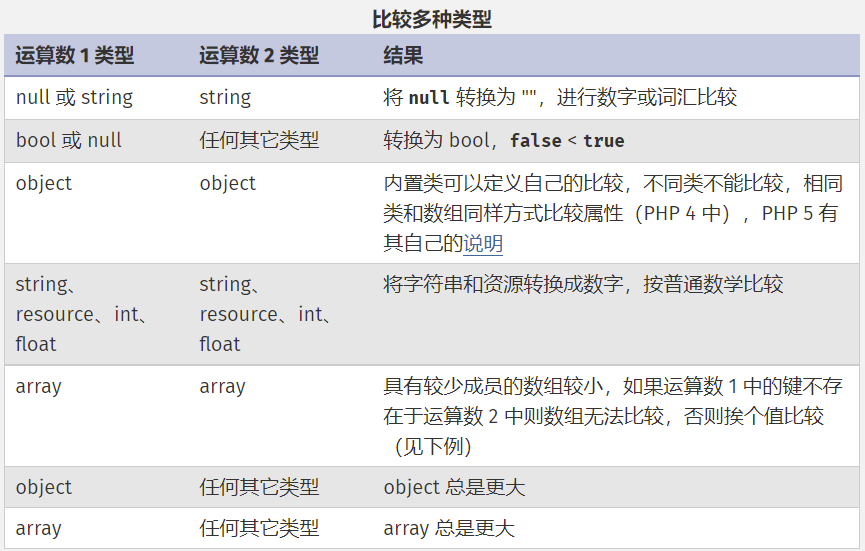
## 三、松散比较符的缺陷

### 理论

php比较相等性的运算符有两种，一种是严格比较，另一种是松散比较。

如果比较一个数字和字符串或者比较涉及到数字内容的字符串，则字符串会被转换成数值并且比较按照数值来进行

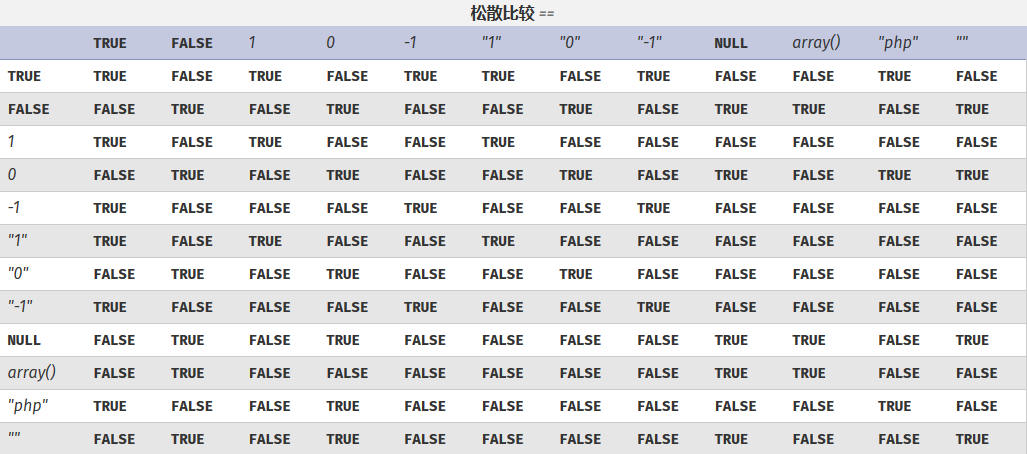
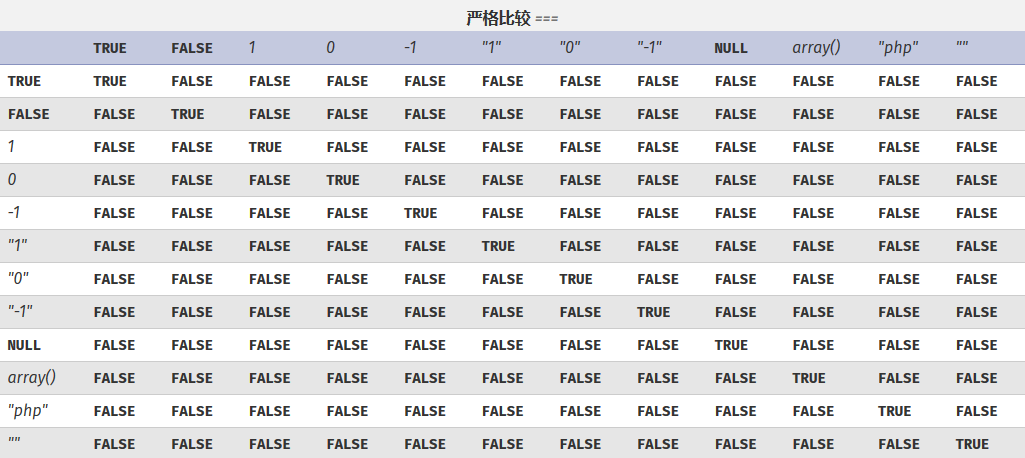




**严格比较符：**会先判断两种字符串的类型是否相等，再比较

=== //全等

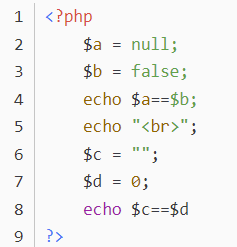
!== //不全等



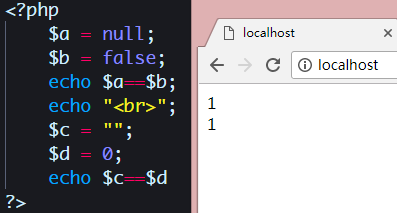
“==”判断数值是否相等，“===”判断数值和类型是否相等

“==”：如果类型不同的进行比较，其会将类型转换成相同的再进行比较

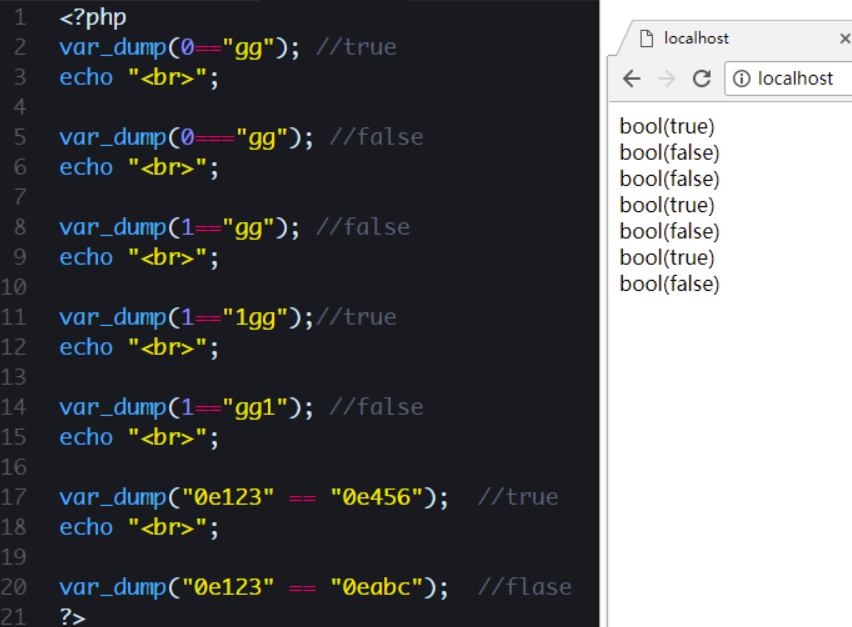
* **例1**：



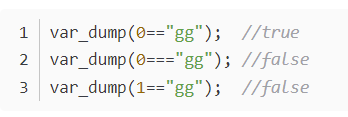
由于php对变量自动转换的特性，这里面的：$a==$b 与 $c==$d 均为真，页面输出结果：



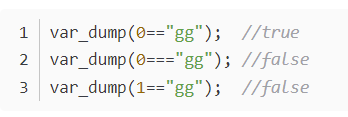
* **例2**：



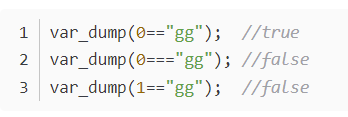
* **结合PHP 相等性比较缺陷解释**：



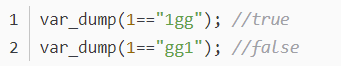
0与gg进行松散性质的不严格比较，会将gg转换为数值，强制转换，由于gg是字符串，转化的结果是0,所以 输出 true



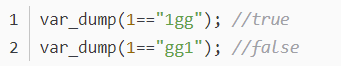
0与gg进行严格性质的严格比较，这里的gg是字符串类型，和int类型的0不相等，所以输出 false



0与gg进行松散性质的不严格比较，会将gg转换为数值，强制转换，由于gg是字符串，转化的结果是0,不等于1，所以输出 false



1与1gg进行松散性质的不严格比较，这里1gg被强制转换为int类型的时候会从字符串的**第一位**开始做判断进行转换，这里的1gg第一位是1，所以这里1gg被转换为1，所以输出 true

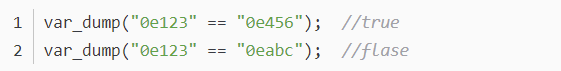


1与gg1进行严格 性质的严格比较，字符串gg1的第一位不是数字，所以它被强制转换为0，所以输出 false

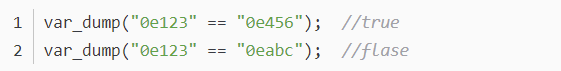
特殊**：**字符串中出现了0e，PHP手册介绍如下:

当一个字符串当作一个数值来取值，其结果和类型如下:

* 如果该字符串没有包含'.','e','E',并且其数值值在整形的范围之内,该字符串被当作int来取值，其他所有情况下都被作为float来取值，该字符串的开始部分决定了它的值
* 如果该字符串以合法的数值开始，则使用该数值，否则其值为0。



0e123与0e456相互不严格性质比较的时候，会将0e这类字符串识为科学技术法的数字,0的无论多少次方都是零，所以相等,输出 true

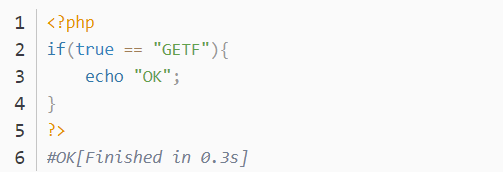


0e123与0eabc相互进行不严格性质比较的时候，本应该将0e这类字符串识为科学技术法的数字,但是这里的0e后面跟着的是abc,数学中科学计数的指数不可以包含字母。所以这里字符串中虽然是0e开头，但是后面的abc却不符合科学技法的规范，所以输出是 false

**例：**



* bool类型的true跟任意字符串可以弱类型相等



* switch同等原理的利用：



### 实践——南京邮电大学网络攻防训md5绕过(Hash比较缺陷)

### **考察点**

### 简单的PHP代码审计

PHP弱类型的Hash比较缺陷

### **题目描述** [md5 collision](http://chinalover.sinaapp.com/web19/" \t "https://blog.csdn.net/qq_27446553/article/details/_blank)****源码****

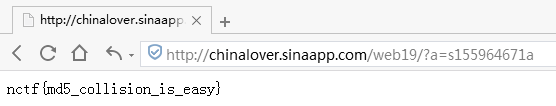


### **write-up**

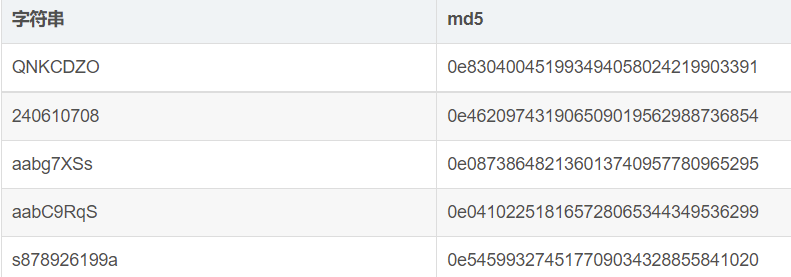
从源码中可以得输入一个a的参数的变量，a首先不等于QNKCDZO并且a的md5值必须等于QNKCDZO加密后的md5值。

乍一看好像不可能存在这样的值，但是这里QNKCDZO加密后的md5值为0e830400451993494058024219903391 这里是0e开头的，在进行等于比较的时候，PHP把它当作科学计数法，0的无论多少次方都是零。 所以这里利用上面的弱类型的比较的缺陷来进行解题：（155964671a和QNKCDZO的md5值都0e开头）

?a=s155964671a



* 字符串加密后md5为0exxxx的字符串(x必须是10进制数字)列表



## 四、sha1() md5()加密函数漏洞缺陷

### 理论

md5()和sha1()对一个数组进行加密（处理数组时）将返回 NULL

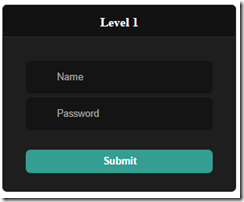
其要求参数不能为数组，那我将传入的参数改成数组，两边return的结果都为false，从而满足不等与相等。将传参变量name，password加个[]即可

### **考察点——**sha1()，md5函数漏洞缺陷

### 实践

* **例1**————****Boston Key Party CTF 2015: Prudential****

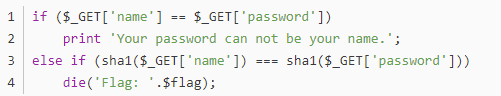
I dont think sha1 is broken.Prove me wrong.



### **write-up**

源代码给出如下:



核心登录代码：

GET类型提交了两个字段name和password，获得flag要求的条件是：

name != password

sha1(name) == sha1(password)

这个乍看起来这是不可能的，但是这里利用sha1()函数在处理数组的时候由于无法处理，将返回NULL可以绕过if语句的验证，if条件成立将获得flag。 构造语句如下:

?name[]=a&password[]=b

这里符合了2个拿到flag的条件：

* a不等于b
* name和password由于是数组，经过sha1()函数后都返回NULL

拿到flag： I\_think\_that\_I\_just\_broke\_sha1

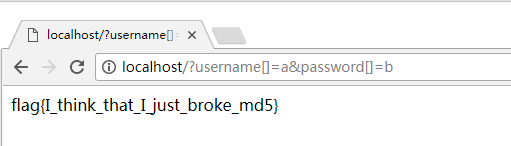
* **例2**

源代码：

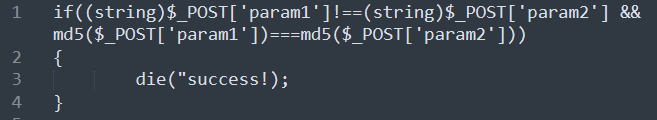


利用md5()函数无法处理数组的这个漏洞，构造GET请求

?username[]=a&password[]=b



* **例3**



Param1=%4d%c9%68%ff%0e%e3%5c%20%95%72%d4%77%7b%72%15%87%d3%6f%a7%b2%1b%dc%56%b7%4a%3d%c0%78%3e%7b%95%18%af%bf%a2%00%a8%28%4b%f3%6e%8e%4b%55%b3%5f%42%75%93%d8%49%67%6d%a0%d1%55%5d%83%60%fb%5f%07%fe%a2

Param2=%4d%c9%68%ff%0e%e3%5c%20%95%72%d4%77%7b%72%15%87%d3%6f%a7%b2%1b%dc%56%b7%4a%3d%c0%78%3e%7b%95%18%af%bf%a2%02%a8%28%4b%f3%6e%8e%4b%55%b3%5f%42%75%93%d8%49%67%6d%a0%d1%d5%5d%83%60%fb%5f%07%fe%a2

MD5值相同使用谷歌可以搜到相当多被巧妙构造出的二进制文件，其MD5相同，注意一点，post时一定要urlencode！！！

## 五、字符串处理函数漏洞缺陷

### 理论

**strcmp()**函数: 比较两个字符串（区分大小写).

**ereg()**函数：字符串正则匹配。指定的模式搜索一个字符串中指定的字符串,如果匹配成功返回true,否则,则返回false。搜索字母的字符是大小写敏感的。

**strpos()**函数：查找字符串在另一字符串中第一次出现的位置，对大小写敏感。

ereg()和strpos()函数都是用来处理字符串的，但是在传入数组参数后都将返回NULL。

* Eregi匹配可以用%00截断
* Eregi匹配可用数组绕过

ereg是处理字符串，传入数组之后，ereg是返回NULL

* **strcmp**用法如下:

int strcmp ( string $str1 , string $str2 )

* 具体的用法解释如下:

参数 `str1`第一个字符串。

参数 `str2`第二个字符串。

如果 `str1` 小于 `str2` 返回 `< 0`；

如果 `str1` 大于 `str2` 返回 `> 0`；

如果两者相等，返回 0。

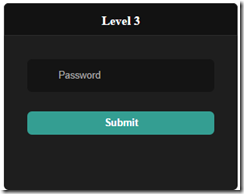
这个函数接受到了不符合的类型，例如数组类型,函数将发生错误。但是在5.3之前的php中，显示了报错的警告信息后，将return 0 !!!! 也就是虽然报错，但却判定其相等了。

### **考察点** 字符串处理函数漏洞缺陷

### 实践 ——Boston Key Party CTF 2015: Northeastern Univ

**题目描述**

## 例1——strcmp（）函数

Of course, a timing attack might be the answer, but Im quite sure that you can do better than that. 

### **write-up**

源代码：



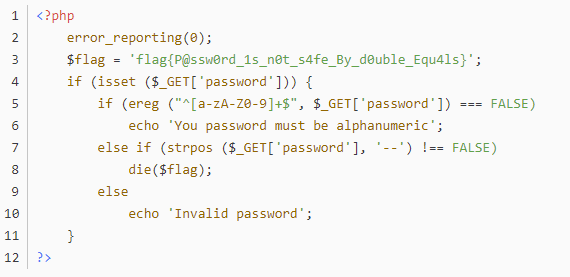
这里使用了==松散比较了$flag和通过GET方式提交的password的值，如果相等的话，拿到flag。这里用的是==松散性质的比较，再利用字符串处理数组时将会报错，在5.3之前的php中，显示了报错的警告信息后，将return 0。

所以这里将password参数指定为数组,利用函数漏洞拿到flag:

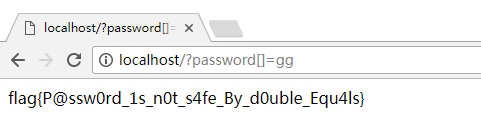
?Password[]=abc

IMG_256

## 例2 ——ereg()和strpos()函数



将参数password赋值数组：http://localhost/?password[]=gg



## 六、变量覆盖漏洞

## 1.parse\_str函数变量覆盖缺陷

### 理论

parse\_str函数的作用就是解析字符串并注册成变量，在注册变量之前不会验证当前变量是否存在，所以直接覆盖掉已有变量。

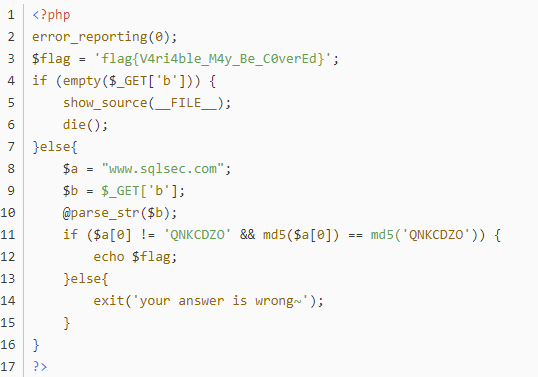
void parse\_str ( string $str [, array &$arr ] )

str 输入的字符串。  
arr 如果设置了第二个变量 arr，变量将会以数组元素的形式存入到这个数组，作为替代。

### **考察点**

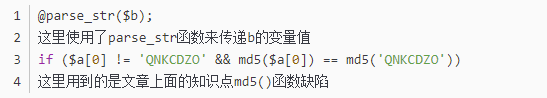
* parse\_str变量覆盖缺陷

### 实践



### **write-up**

找到核心代码:

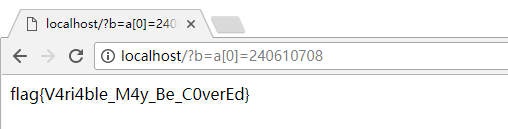


因为这里用到了parse\_str函数来传递b，if的语句的条件是拿$a[0]来比较的，因为这里的变量a的值已经三是固定的了:

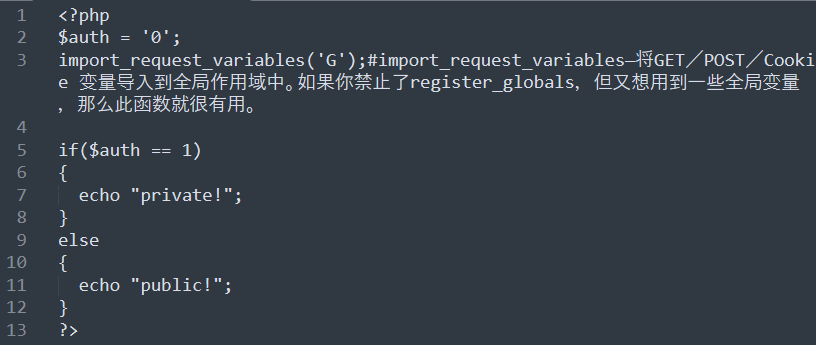
$a = "www.sqlsec.com";

利用变量覆盖函数的缺陷：这里可以对a的变量进行重新赋值，后面的的if语句再利用本文前面提到的md5()比较缺陷进行绕过:

<http://localhost/?b=a[0]=240610708>

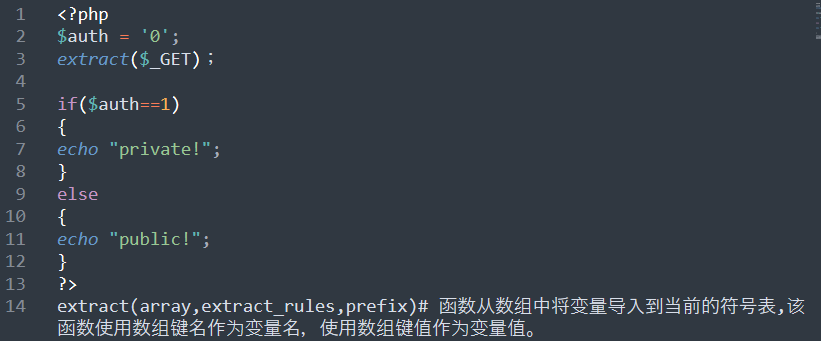


**2.import\_request\_variables变量覆盖**



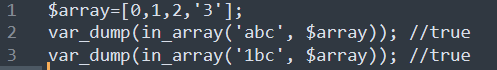
当用户访问链接为www.xxx.com/test.php?auth=aaa时就会出现变量覆盖问题

### **extract()变量覆盖**



当用户访问链接为www.xxx.com/test.php?auth=aaa时就会出现变量覆盖问题，安全的做法是确定register\_globals=OFF后，在调用extract()时使用EXTR\_SKIP保证已有变量不会被覆盖。

## 七．in\_array()强转类型



**注意**：在所有php认为是int的地方输入string，都会被强制转换

参考文献：

[PHP代码安全杂谈 | 国光 (sqlsec.com)](https://www.sqlsec.com/2018/01/php.html)

<http://php.net/manual/zh/language.operators.comparison.php>

<http://php.net/manual/zh/language.types.float.php>

<http://php.net/manual/zh/types.comparisons.php>

<https://www.cnblogs.com/Mrsm1th/p/6745532.html>

<https://blog.csdn.net/qq_35078631/article/details/62980648>

<https://www.cnblogs.com/xishaonian/p/7628152.html>

<https://blog.csdn.net/cherrie007/article/details/77473817>

<https://www.jianshu.com/p/e7cf997d6ccb>

<https://www.cnblogs.com/sqyysec/p/6926095.html>

[PHP代码安全杂谈\_If you can take it, you can make it.-CSDN博客](https://blog.csdn.net/qq_27446553/article/details/79270812)

[PHP代码安全【PHP弱口令、加密函数、绕过函数】/CTF代码审计题\_Sp4rkW的博客-CSDN博客](https://blog.csdn.net/wy_97/article/details/79088218)