

php bapss弱类型,PHP代码安全杂谈!

转载

[weixin_39686230](#) 于 2021-04-02 05:13:51 发布 35 收藏

文章标签: [php bapss弱类型](#)

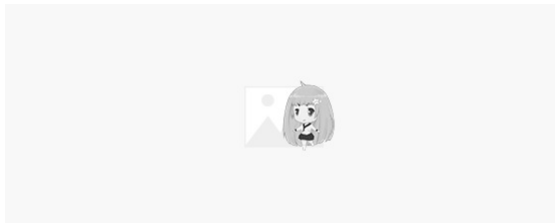
摘要:

虽然PHP是世界上最好的语言,但是也有一些因为弱类型语言的安全性问题出现。WordPress历史上就出现过由于PHP本身的缺陷而造成的一些安全性问题,如CVE-2014-0166中的cookie伪造就是利用了PHP Hash比较的缺陷。当然一般这种情况实战中用到的不是很多,但是在CTF竞赛中却是一个值得去考察的一个知识点,特此记录总结之。

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。下面看一个有趣的例子:

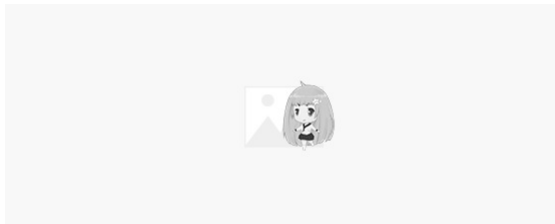


以十进制能够精确表示的有理数如 0.1 或 0.7,无论有多少尾数都不能被内部所使用的二进制精确表示,因此不能在不丢失一点点精度的情况下转换为二进制的格式。这就会造成混乱的结果:例如, $\text{floor}((0.1+0.7)*10)$ 通常会返回 7 而不是预期中的 8,因为该结果内部的表示其实是类似 $7.9999999999999991118\dots$ 。

实践

问鼎杯2017 老眼昏花网上很多write-up感觉就像是看着答案写write-up,个人感觉真正的write-up中应该体现自己的思考在里面。

题目描述

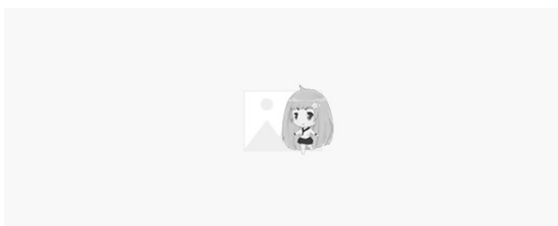


题目言简意赅,让我们把2017这个值传递给服务器。

考察点PHP浮点精确度

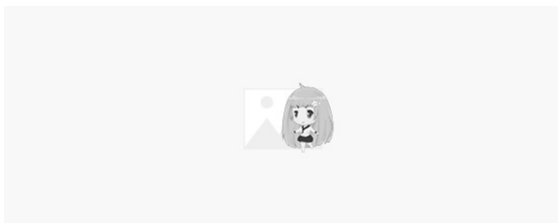
write-up

what year is this?所以第一反应是直接给year参数赋值为2017: ?year=2017

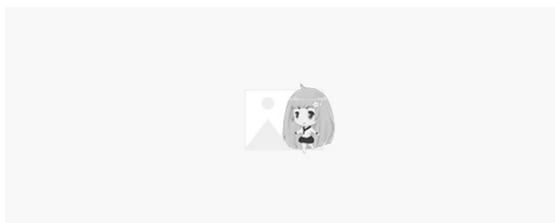


然而结果如下:

有提示了,说明year这个参数是对的,但是2017中不可以出现7,这里如果不了解php精度的话,肯定是对2017进行各种编码绕过,但是这里对编码也进行过滤了:



所以最后一种可能就是利用PHP精度来绕过: ?year=2016.999999999999



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。

实践

is_numeric()用于判断是否是数字，通常配合数值判断。

案例代码<?php

```
error_reporting(0);
```

```
$flag = 'flag{1S_umer1c_Not_S4fe}';
```

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

```
is_numeric($id)?die("Sorry..."):NULL;
```

```
if($id>665){ echo $flag;
```

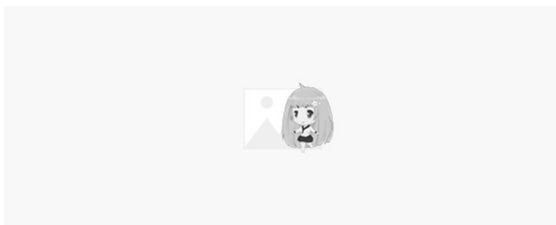
```
}
```

```
?>
```

考察点PHP类型转换缺陷

write-up

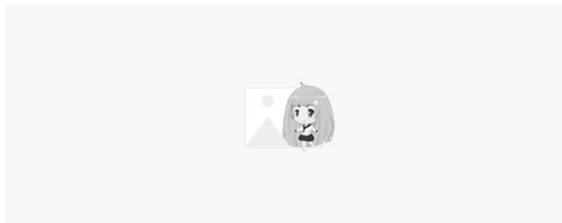
分析下代码:首先对GET方式提交的参数id的值进行检验。id通过is_numeric函数来判断是否为数字，如果是数字的话，GG。如果不是数字的话，和665进行比较，id的值大于665的时候输出flag。乍看上去又好像不可能这里，但是如果知道PHP弱类型语言的一个特性，当一个整形和一个其他类型行比较的时候，会先把其他类型intval数字化再比。这个特性的话就可以很好的绕过。http://localhost/?id=666gg



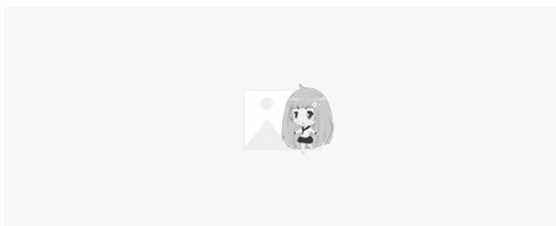
三、松散比较符的缺陷

理论

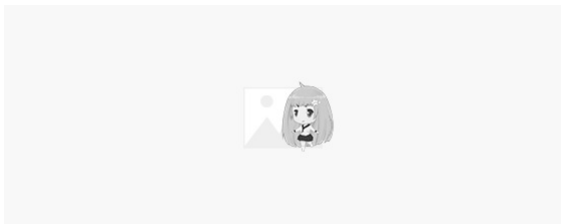
php比较相等性的运算符有两种，一种是严格比较，另一种是松散比较。如果比较一个数字和字符串或者比较涉及到数字内容的字符串，则字符串会被转换成数值并且比较按照数值来进行



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



松散比较符松散比较符，会先将字符串类型转换成相同，再比较。== //等于!= //不等



PHP会根据变量的值，自动把变量转换为正确的数据类型。这一点和C和C++以及Java之类的语言明显不同。虽然这样PHP方便了程序员，但是随之而来却会带来一些安全性的问题。

一个简单的例子<?php

```
$a = null;
```

```
$b = false; echo $a==$b; echo "
```

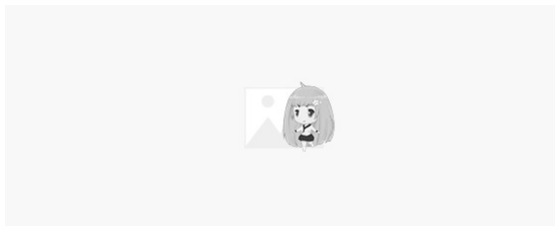
```
";
```

```
$c = "";
```

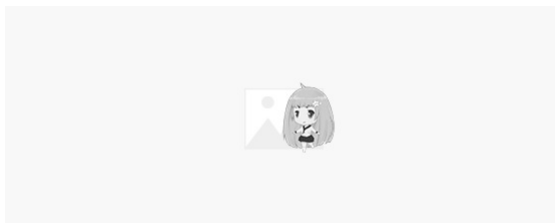
```
$d = 0; echo $c==$d?>
```

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

所以页面输出的结果为:



一个深入的例子



下面结合PHP 相等性比较缺陷再解释下会好懂一点:`var_dump(0=="gg"); //true``var_dump(0==="gg"); //false``var_dump(1=="gg"); //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,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``var_dump(1=="1gg"); //true``var_dump(1=="gg1"); //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``var_dump("0e123" == "0e456"); //true``var_dump("0e123" == "0eabc"); //false`

这里比较特殊, 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不符合科学技术的规范, 所以输出是 false

实践

md5绕过(Hash比较缺陷)南京邮电大学网络攻防训练平台中一道比较经典的md5 collision题, 关于这道题目的WriteUp网上很多, 但是真正深入分析的少之又少~~

题目描述<?php

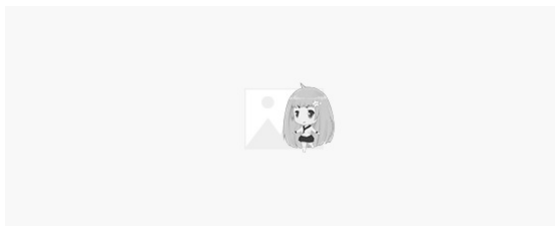
```
$md51 = md5('QNKCDZO');
$a = @$_GET['a'];
$md52 = @md5($a); if(isset($a)){ if ($a != 'QNKCDZO' && $md51 == $md52) { echo "nctf{*****}";
} else { echo "false!!!";
}
} else{ echo "please input a";
}?>
```

考察点简单的PHP代码审计

PHP弱类型的Hash比较缺陷

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得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: ?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

字符串md5

QNKCDZO0e830400451993494058024219903391

2406107080e462097431906509019562988736854

aabg7Xs0e087386482136013740957780965295

aabC9RqS0e041022518165728065344349536299

s878926199a0e545993274517709034328855841020

四、sha1() md5()加密函数漏洞缺陷

理论

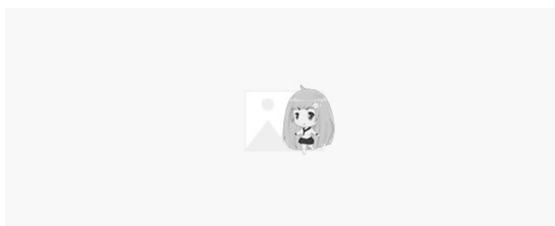
md5()和sha1()对一个数组进行加密将返回 NULL

实践

Boston Key Party CTF 2015: Prudential

题目描述I dont think sha1 isbroken.Prove me wrong.

题目给了一个登陆框:



考察点sha1()函数漏洞缺陷

write-up

源代码给出如下:

```
level1<?phprequire
'flag.php';if (isset($_GET['name']) and isset($_GET['password'])) { if ($_GET['name'] == $_GET['password']) print 'Your password can not be your name.'; else if (sha1($_GET['name']) === sha1($_GET
print '
Invalid password.
';
}>
```

Invalid password.

;

Level 1

分析一下核心登录代码如

```
下:if ($_GET['name'] == $_GET['password']) print 'Your password can not be your name.';else if (sha1($_GET['name']) === sha1($_GET['password'])) die('Flag: '.$flag);
```

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, 如果条件成立将获得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()函数嫁给后都返回NULL

拿到flag: I_think_that_I_just_broke_sha1

拓展总结

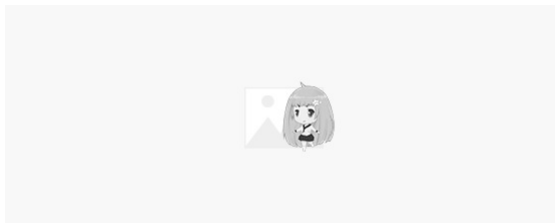
经过验证, 不仅sha1()函数无法处理数组, 这里md5()函数也有同样的问题, 在处理数组的时候, 都将返回NULL测试代码如下:<?phperror_reporting (0);

```
$flag = 'flag{I_think_that_I_just_broke_md5}';if (isset($_GET['username']) and isset($_GET['password'])) { if ($_GET['username'] == $_GET['password']) print 'Your password
print 'Invalid password';
}>
```

这里面的核心代码如下:if (\$_GET['username'] == \$_GET['password'])

并且得满足:if (md5(\$_GET['username']) === sha1(\$_GET['password']))

同样利用md5()函数无法处理数组的这个漏洞, 构造get请求拿到flag:?username[]=a&password[]=b



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).

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 `<

如果 `str1` 大于 `str2` 返回 `>

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 !!!! 也就是虽然报了错, 但却判定其相等了。

ereg()函数: 字符串正则匹配。

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。

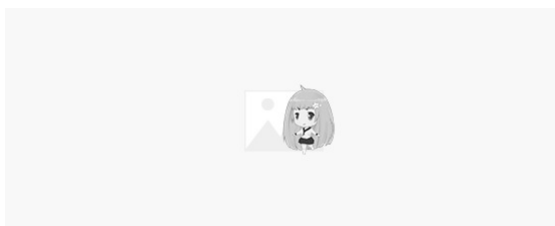
这2个函数都是用来处理字符串的, 但是在传入数组参数后都将返回NULL。

实践

Boston Key Party CTF 2015: Northeastern Univ

题目描述

Of course, a timing attack might be the answer, but Im quite sure that you can do better than that. 题目给了一个登陆框:



考察点字符串处理函数漏洞缺陷

write-up

给出源代码如下:

```
level3<?phprequire 'flag.php';if (isset($_GET['password'])) { if (strcmp($_GET['password'], $flag) == 0) die('Flag: '.$flag); else print 'Invalid password.'
```

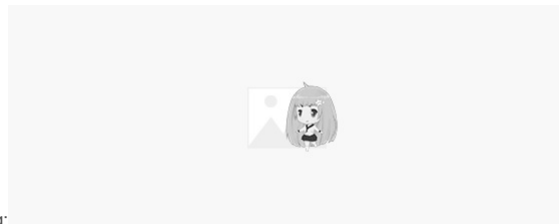
```
};
```

```
}?>
```

Level 3

分析一下核心登录代码如下:if (strcmp(\$_GET['password'], \$flag) == 0)

这里使用了==松散比较了\$flag和通过GET方式提交的password的值, 如果想等的话, 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。所有这里将password参数指定为数组,利用函数漏洞拿到flag:

拓展总结

除了strcmp()函数外, ereg()和strpos()函数在处理数组的时候也会异常, 返回NULL。测试代码如下:<?php

```
error_reporting(0);
```

```
$flag = 'flag{P@ssw0rd_1s_n0t_s4fe_By_d0uble_Equ4ls}'; if (isset($_GET['password'])) {
```

```
if (ereg ("^[a-zA-Z0-9]+$", $_GET['password']) === FALSE)
```

```
echo 'You password must be alphanumeric';
```

```
else if (strpos ($_GET['password'], '-') !== FALSE)
```

```
die($flag);
```

```
else
```

```
echo 'Invalid password';
```

```
}
```

```
?>
```

将参数password赋值一个数组传递进去: http://localhost/?password[]=gg

ereg()函数是处理字符串的，传入数组后返回NULL，NULL和FALSE，是不恒等(==)的，满足第一个if条件；而strpos()函数也是处理字符串的，传入数组后返回



NULL，NULL!==FALSE，满足条件，拿到flag:

六、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。

实践

测试代码:<?phperror_reporting (0);

```
$flag = 'flag{V4ri4ble_M4y_Be_C0verEd}';if (empty($_GET['b'])) {
show_source(__FILE__); die();
}else{
$a = "www.sqlsec.com";
$b = $_GET['b'];
@parse_str($b); if ($a[0] != 'QNKCDZO' && md5($a[0]) == md5('QNKCDZO')) { echo $flag;
}else{ exit('your answer is wrong-');
}
}
}?
```

考察点parse_str变量覆盖缺陷

write-up

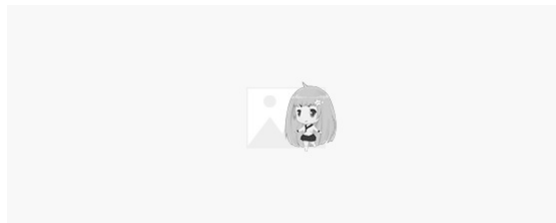
找到核心代码:@parse_str(\$b);

这里使用了parse_str函数来传递b的变量值if (\$a[0] != 'QNKCDZO' && md5(\$a[0]) == md5('QNKCDZO'))

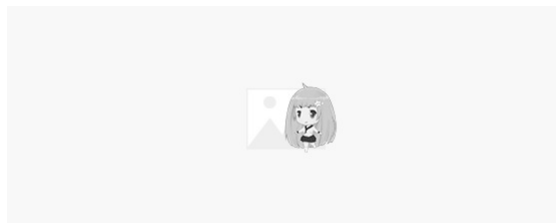
这里用到的是文章上面的知识点md5()函数缺陷

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，有因为这里的变量a的值已经三是固定的了:\$a = "www.sqlsec.com";

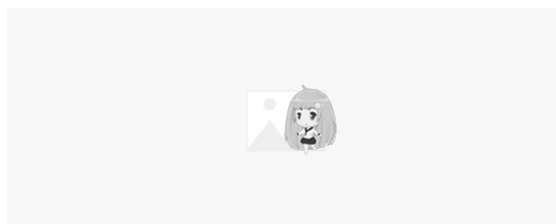
这里其实是我博客的地址~~ 不过不重要。整体代码乍看起来又不可能，但是利用变量覆盖函数的缺陷这里可以对a的变量进行重新赋值，后面的的if语句再利用本文前面提到的md5()比较缺陷进行绕过:http://localhost/?b=a[0]=240610708



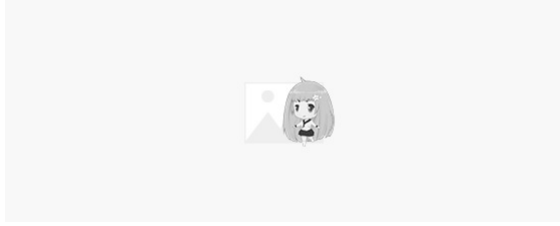
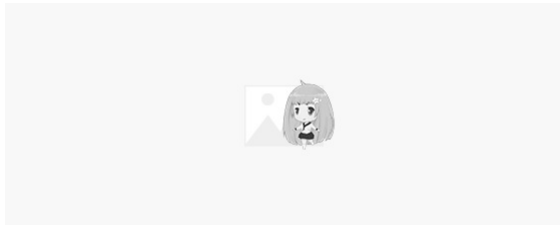
参考文献



感谢您的支持，我会继续努力的!



扫码打赏，你说多少就多少



打开支付宝扫一扫，即可进行扫码打赏哦