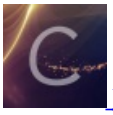
CTF中常见的PHP函数漏洞

原创

Hacking黑白红 于 2020-07-04 12:08:45 发布 1438 收藏 3
分类专栏： bugku CTF 信息安全 文章标签： 安全

本文链接：https://blog.csdn.net/zsw15841822890/article/details/107122486


bugku 同时被 3 个专栏收录
7 篇文章 2 订阅
订阅专栏


CTF
15 篇文章 6 订阅
订阅专栏


信息安全
39 篇文章 8 订阅
订阅专栏

1.弱类型比较

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

drops.wooyun.org

2.MD5 compare漏洞

PHP在处理哈希字符串时，如果利用"!="或"=="来对哈希值进行比较，它把每一个以"0x"开头的哈希值都解释为科学计数法0的多少次方（为0），所以如果两个不同的密码经过哈希以后，其哈希值都是以"0e"开头的，那么php将会认为他们相同。

常见的payload有

0x01 md5(str)

```
QNKCDZO

240610708

s878926199a

s155964671a

s214587387a

s214587387a
```

0x02 sha1(str)

```
sha1('aaroZmOk')

sha1('aaK1STfY')

sha1('aaO8zKZF')

sha1('aa3OFF9m')
```

同时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://127.0.0.1/1.php?a[]=1&b[]=2
```

现在更厉害了…自从王小云教授提出了MD5碰撞之后这个就成了大热门，现在网上流传一个诸多密码专家写的MD5碰撞程序，是根据一个文件，然后填充内容生成两个MD5值一样的文件（有一定失败率其实），然后生成的内容MD5就相同了…强网杯签到题，简直震惊…软件是fastcoll_v1.0.0.5，自行下载吧

3.ereg函数漏洞：00截断

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

字符串对比解析

在这里如果 $_GET[‘password’]为数组，则返回值为NULL ， null!=false

如果为123 || asd || 12as || 123%00&&&**，则返回值为true ,%00直接截断

其余为false

ereg()区分大小写，eregi()不区分大小写

4.$key是什么？

别忘记程序可以把变量本身的key也当变量提取给函数处理。

```
<?php

    print_r(@$_GET);

    foreach ($_GET AS $key => $value)

    {

        print $key."\n";

    }

?>
```

5.变量覆盖

主要涉及到的函数为extract函数，看个例子

```
<?php

    $auth = '0';

    // 这里可以覆盖$auth的变量值

    print_r($_GET);

    echo "</br>";

    extract($_GET);

    if($auth == 1){

        echo "private!";

    } else{

        echo "public!";

    }

?>
```

extract可以接收数组，然后重新给变量赋值，过程页很简单。

这里写图片描述

同时！PHP的特性$可以用来赋值变量名也能导致变量覆盖！

```php
<?php
    $a='hi';
    foreach($_GET as $key => $value) {
        echo $key."</br>".$value;
        $$key = $value;
    }
    print "</br>".$a;
?>
```

构造http://127.0.0.1:8080/test.php?a=12 即可达到目的。

6.strcmp

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
<?php
    $password=$_GET['password'];
    if (strcmp('xd',$password)) {
     echo 'NO!';
    } else{
        echo 'YES!';
    }
?>
```

构造http://127.0.0.1:8080/test.php?password[]=

7.is_numeric

无需多言：

```
<?php

echo is_numeric(233333);       # 1

echo is_numeric('233333');    # 1

echo is_numeric(0x233333);    # 1

echo is_numeric('0x233333');   # 1

echo is_numeric('233333abc');  # 0

?>
```

8.preg_match

如果在进行正则表达式匹配的时候，没有限制字符串的开始和结束(^ 和 $)，则可以存在绕过的问题

```
<?php

$ip = 'asd 1.1.1.1 abcd'; // 可以绕过

if(!preg_match("/(\d+)\.(\d+)\.(\d+)\.(\d+)/",$ip)) {

  die('error');

} else {

   echo('key...');

}

?>
```

9.parse_str

与 parse_str() 类似的函数还有 mb_parse_str()，parse_str 将字符串解析成多个变量，如果参数str是URL传递入的查询字符串（query string），则将它解析为变量并设置到当前作用域。

时变量覆盖的一种

```php
<?php

    $var='init';

    print $var."</br>";

    parse_str($_SERVER['QUERY_STRING']);

    echo $_SERVER['QUERY_STRING']."</br>";

    print $var;

?>
```

10.字符串比较

```php
<?php

    echo 0 == 'a' ;// a 转换为数字为 0    重点注意

    // 0x 开头会被当成16进制54975581388的16进制为 0xccccccccc

    // 十六进制与整数，被转换为同一进制比较

    '0xccccccccc' == '54975581388' ;



    // 字符串在与数字比较前会自动转换为数字，如果不能转换为数字会变成0

    1 == '1';

    1 == '01';

    10 == '1e1';

    '100' == '1e2' ;



    // 十六进制数与带空格十六进制数，被转换为十六进制整数

    '0xABCdef'  == '     0xABCdef';

    echo '0010e2' == '1e3';

    // 0e 开头会被当成数字，又是等于 0*10^xxx=0

    // 如果 md5 是以 0e 开头，在做比较的时候，可以用这种方法绕过

    '0e509367213418206700842008763514' == '0e481036490867661113260034900752';

    '0e481036490867661113260034900752' == '0' ;



    var_dump(md5('240610708') == md5('QNKCDZO'));

    var_dump(md5('aabg7XSs') == md5('aabC9RqS'));

    var_dump(sha1('aaroZmOk') == sha1('aaK1STfY'));

    var_dump(sha1('aaO8zKZF') == sha1('aa3OFF9m'));

?>
```

11.unset

unset(bar);用来销毁指定的变量，如果变量bar 包含在请求参数中，可能出现销毁一些变量而实现程序逻辑绕过。

```php
<?php

$_CONFIG['extraSecure'] = true;

foreach(array('_GET','_POST') as $method) {

    foreach($$method as $key=>$value) {

      // $key == _CONFIG

      // $$key == $_CONFIG

      // 这个函数会把 $_CONFIG 变量销毁

      unset($$key);

    }

}

if ($_CONFIG['extraSecure'] == false) {

    echo 'flag {****}';

}

?>
```

12.intval()

int转string：

$var = 5;

方式1：$item = (string)$var;

方式2：$item = strval($var);

string转int：intval()函数。

var_dump(intval('2')) //2

var_dump(intval('3abcd')) //3

var_dump(intval('abcd')) //0

可以使用字符串-0转换，来自于wechall的方法

说明intval()转换的时候，会将从字符串的开始进行转换直到遇到一个非数字的字符。即使出现无法转换的字符串，intval()不会报错而是返回0

顺便说一下，intval可以被%00截断

```
if($req['number']!=strval(intval($req['number']))){

     $info = "number must be equal to it's integer!! ";

}
```

如果当$req[‘number’]=0%00即可绕过

13.switch()

如果switch是数字类型的case的判断时，switch会将其中的参数转换为int类型，效果相当于intval函数。如下：

```
<?php
   $i ="abc";
   switch ($i) {
   case 0:
   case 1:
   case 2:
   echo "i is less than 3 but not negative";
   break;
   case 3:
   echo "i is 3";
   }
?>
```

14.in_array()

```
$array=[0,1,2,'3'];
var_dump(in_array('abc', $array)); //true
var_dump(in_array('1bc', $array)); //true
```

在所有php认为是int的地方输入string，都会被强制转换

15.serialize 和 unserialize漏洞

这里我们先简单介绍一下php中的魔术方法（这里如果对于类、对象、方法不熟的先去学学吧），即Magic方法，php类可能会包含一些特殊的函数叫magic函数，magic函数命名是以符号__开头的，比如 __construct，__destruct，__toString，__sleep，__wakeup等等。这些函数都会在某些特殊时候被自动调用。

例如__construct()方法会在一个对象被创建时自动调用，对应的__destruct则会在一个对象被销毁时调用等等。

这里有两个比较特别的Magic方法，__sleep 方法会在一个对象被序列化的时候调用。 __wakeup方法会在一个对象被反序列化的时候调用。

```
<?php

class test

{

    public $username = '';

    public $password = '';

    public $file = '';

    public function out(){

        echo "username: ".$this->username."<br>"."password: ".$this->password ;

    }

     public function __toString() {

        return file_get_contents($this->file);

    }

}

$a = new test();

$a->file = 'C:\Users\YZ\Desktop\plan.txt';

echo serialize($a);

?>
```

//tostring方法会在输出实例的时候执行，如果实例路径是隐秘文件就可以读取了

echo unserialize触发了__tostring函数，下面就可以读取了C:\Users\YZ\Desktop\plan.txt文件了

```php
<?php

class test

{

    public $username = '';

    public $password = '';

    public $file = '';

    public function out(){

        echo "username: ".$this->username."<br>"."password: ".$this->password ;

    }

     public function __toString() {

        return file_get_contents($this->file);

    }

}

$a = 'O:4:"test":3:{s:8:"username";s:0:"";s:8:"password";s:0:"";s:4:"file";s:28:"C:\Users\YZ\Desktop\plan.t

echo unserialize($a);

?>
```

16.session 反序列化漏洞

主要原因是

ini_set('session.serialize_handler', 'php_serialize');

ini_set('session.serialize_handler', 'php');
两者处理session的方式不同

【注】参考https://www.cnblogs.com/ningskyer/articles/9328337.html