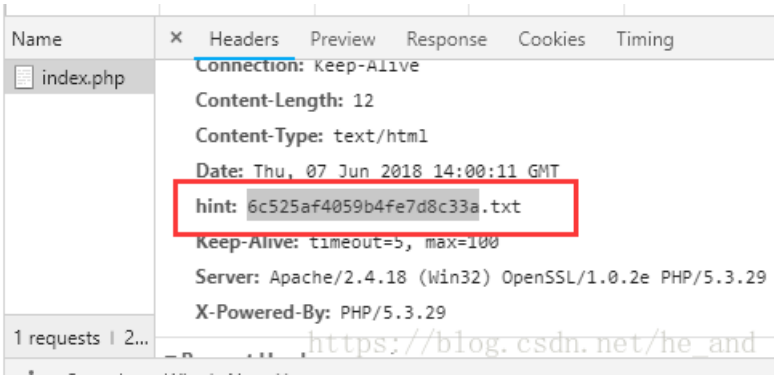
php intval溢出,实验吧-你真的会PHP吗

转载

Rita201907 于 2021-03-16 16:23:29 发布 81 收藏
文章标签： php intval溢出
根据题目提示可以知道这是一道代码审计的题目，所以就找找源码放哪里的吧，最终在响应头中找到了。



访问的到源码：

<?php $info = ""; $req = []; $flag="xxxxxxxxxx"; ini_set("display_error", false); error_reporting(0); if(!isset($_POST['number'])){ header("hint:6c525af4059b4fe7d8c33a.txt"); die("have a fun!!"); } foreach([$_POST] as $global_var) { foreach($global_var as $key => $value) { $value = trim($value); is_string($value) && $req[$key] = addslashes($value); } } function is_palindrome_number($number) { $number = strval($number); $i = 0; $j = strlen($number) - 1; while($i < $j) { if($number[$i] !== $number[$j]) { return false; } $i++; $j--; } return true; } if(is_numeric($_REQUEST['number'])){ $info="sorry, you cann't input a number!"; }elseif($req['number']!=strval(intval($req['number']))){ $info = "number must be equal to it's integer!! "; }else{ $value1 = intval($req["number"]); $value2 = intval(strrev($req["number"])); if($value1!=$value2){ $info="no, this is not a palindrome number!"; }else{ if(is_palindrome_number($req["number"])){ $info = "nice! {$value1} is a palindrome number!"; }else{ $info=$flag; } } } echo $info;

我们分段来看源码的功能：

foreach([$_POST] as $global_var) {

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

$value = trim($value);

is_string($value) && $req[$key] = addslashes($value);

}

}

上面这一段很明显就是需要post传送过来的数据是一个字符串，是字符串才会存放到$req数组中。

function is_palindrome_number($number) {

$number = strval($number);

$i = 0;

$j = strlen($number) - 1;

```
while($i < $j) {
if($number[$i] !== $number[$j]) {
return false;
}
$i++;
$j--;
}
return true;
}
```

该函数判断输入的数是否是一个回文数。

接下来的一段：

我看writeup看到有两种解法：

利用intval溢出

intval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('1000000000000') 会返回 2147483647。64 位系统上，最大带符号的 integer 值是 9223372036854775807。

通过上面我们知道服务器的操作系统是32位的，所以我们构造2147483647就可以同时满足2，3条件。通过把空字符可以绕过is_numeric的判断(如%00,%20),所以我们构造以下poc，number=2147483647%00 和 number=2147483647%20都可。

我们来看上面的payload是怎么绕过上面的条件的，首先因为我们post的number中包含%00或者%20这样的空字符，所以在is_numeric判断时，会返回false,接下来

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

intval会忽略掉我们的空字符%00与%20，所以这里也就绕过了，然后：

$value1 = intval($req["number"]);

$value2 = intval(strrev($req["number"]));

这两个值要相等，由于我们输入的number已经达到了intval的最大值，所以当执行strrev后，得到7463847412这个值，这个值经过intval转换为2147483647，所以这两个值相等了。

但是2147483647又不是回文数，所以得到flag。

注：strval会忽略掉%00与%20

注：如果你输入number='1'这样的字符，后台存储的字符串时'\'1\'"，意思就是会把引号作为你输入的字符串的一部分。这是个很奇怪的特性，大家可以测试一下

注：其中很多细节，我是通过把源码拷贝到本地执行得到的结果，大家也可以测试一下自己的想法。

0x02科学记数法绕过

因为要求不能为回文数，但又要满足intval(req["number"])=intval(strrev(r

e

q

[

"

n

u

m

b

e

r

"

]

)

=

i

n

t

v

a

l

(

s

t

r

r

e

v

(req[“number”]))，所以我们采用科学计数法构造poc为number=0e-0%00，这样的话我们就可以绕过。

一定要时-0，才不会被判定为回文数

不得不说脑洞是真的大

积硅步，致千里