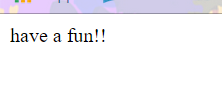
实验吧做的一道题

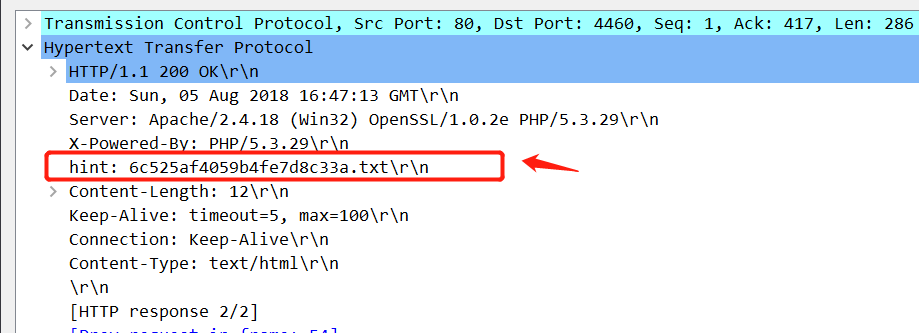
题目链接：

<http://ctf5.shiyanbar.com/web/PHP/index.php>

打开只是很简单的页面，原码也没什么可以利用的信息



没办法，用burpsuit抓个包，发送给Reapeater。



可以看到，有一个txt文件，然后改了 一下网页链接，把这文件打开来看看

Emmmm。。。代码审计啊？

**<?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!!");**

**}**

**//遍历**

**foreach([$\_POST] as $global\_var) {**

**foreach($global\_var as $key => $value) {**

**$value = trim($value); //trim() 函数移除字符串两侧的空白字符或其他预定义字符。**

**is\_string($value) && $req[$key] = addslashes($value);**

**}**

**}**

**//global $var是外部$var的同名引用或者指针。**

**//函数**

**function is\_palindrome\_number($number) {**

**$number = strval($number); //本函数可将数组及类之外的变量类型转换成字符串类型。**

**$i = 0;**

**$j = strlen($number) - 1;//strlen() 函数返回字符串的长度**

**while($i < $j) {**

**if($number[$i] !== $number[$j]) {**

**return false;**

**}**

**$i++;**

**$j--;**

**}**

**return true;**

**}**

**//判断是否为数值型**

**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"])); //strrev() 函数反转字符串。**

**if($value1!=$value2){**

**$info="no, this is not a palindrome number!";**

**}else{**

**//判断回文数**

**if(is\_palindrome\_number($req["number"])){**

**$info = "nice! {$value1} is a palindrome number!";**

**}else{**

**$info=$flag;**

**}**

**}**

**}**

**echo $info;**

经过审计，得出一下Post的三个传递限制条件：

1.不为空，且不能是一个数值型数字，包括小数。(由is\_numeric函数判断)

2.不能是一个回文数。（is\_palindrome\_number判断；回文数就是类似于121这样的数。）

3.该数的反转的整数值应该和它本身的整数值相等。

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。

**在php中的三种溢出形式**

***<?php  
echo intval(1e10)."<br />";                    // 1410065408  
echo intval(420000000000000000000)."<br />";   // 0  
echo intval('420000000000000000000')."<br />"; // 2147483647  
?>***

回复于 2009-08-28 17:12:22 #8 得分：20

1.intval(1e10)

其实就是intval(10000000000);即10的10次方

对于这种类型的转换，由于他已经超过的int的范围了，在进行intval转 换时，

他其实只取出来整型的那部分，然后选截取整型（32位）的大小，再转成int型，

这种情况下，假设1e10转 成二进制为：1001010100000010111110010000000000

那他截取32位后就变成01010100000010111110010000000000 然后去掉第一位就是：

1010100000010111110010000000000 就是他所表示的正的整型数（第一位为0）

像如intval(1e12)的值转成二进制的是：1110100011010100101001010001000000000000

取32位是：11010100101001010001000000000000

那转换成整型就是1010100101001010001000000000000－＞727379968

最高位是1所以结果就是 -727379968

2.intval(420000000000000000301)

这种情况下，则于参数内实际是保存了double型的（php内为float）这种转换是溢出了，直接返回0，在linux下面有时返回－2147483648－不难理解！

3.echo intval('420000000000000300000')

这种情况其实根第一种有点相似，里面的参数其实在内存内是以字符格式保存的（第一种是以float型式存储的），在转换成整型时，由于字符串地址与整型的地址存储方式不一样，对于转换整型的intval函数去按地址读出来后，其实读出来的只是数值型的那部分，高出的都去掉了，也就是达到了int的最大的值，但因为是直接读取地址的，把字符串的地址当整型来读，读进来的就是整型，所以也不会出现溢出的情况，所以得到的其实就是转换成int的最大值。相当于最高位其实是0后面都是1，所以就是2的31次方式减1，就是你输出来的值。

**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(req["number"])=intval(strrev(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