目录

[一、基础介绍 1](#_Toc8718)

[二、魔术方法 4](#_Toc17743)

[·\_\_wakeup 5](#_Toc29409)

[·\_\_construct 5](#_Toc13723)

[·\_\_destruct 6](#_Toc5840)

[·\_\_call 7](#_Toc5112)

[·\_\_get 8](#_Toc9047)

[·\_\_toString 9](#_Toc25894)

[·\_\_invoke 10](#_Toc675)

[三、PHP反序列化考点 11](#_Toc10593)

[①\_\_wakeup绕过 11](#_Toc32698)

[②Bypass反序列化正则绕过 12](#_Toc1125)

[③PHP原生类的利用 12](#_Toc26864)

[1) Exception类（\_\_tostring方法） 12](#_Toc1690)

[2) Error类（\_\_tostring方法） 13](#_Toc8937)

[3) SimpleXMLElement类（\_\_tostring方法） 16](#_Toc10377)

[4) SoapClient类（\_\_call方法） 16](#_Toc6321)

[④phar反序列化 16](#_Toc21257)

[(1) phar简介 16](#_Toc6594)

[(2) phar文件结构 17](#_Toc25030)

**一、基础介绍**

要了解反序列化，先要了解序列化是什么东西。

序列化就是将数据转换为一种可逆的数据结构，逆向的过程就称为反序列化。

* 举个栗子：你在电视上看到一个乐高拼成的船，然后你下单买过来，商家是先把船拆分成各个小部件，再给你寄过来，拆分成小部件的过程称为序列化；你收到这些小部件后，再将他们拼回去，拼成船，这个拼回去的过程称为反序列化。

在php中，序列化和反序列化函数分别为serialize和unserialize

了解序列化之前，我们需要了解php中的类和对象

类：实现某个功能的集合 （class 类）

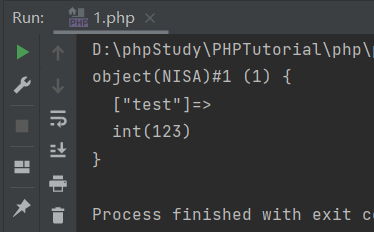
对象：集合中的某个具体的东西 （new 对象）

* 举个栗子：人类为一个类的话，那小明就是一个对象，小红小刚也是；如果汽车为一个类的话，那小明家的汽车就是一个对象。

代码：

class NISA{ //class一个类名  
 var $test = 123; //var声明定义一个变量  
}  
$aaa = new NISA ; //new一个对象  
var\_dump($aaa); var\_dump输出对象

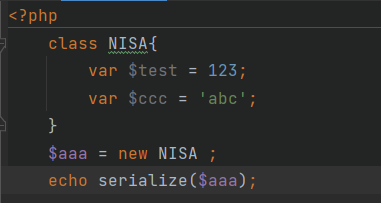
输出结果为：



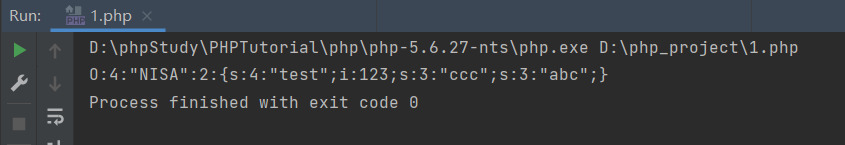
类当中不止可以放变量，还可以放常量、函数，函数在类中又叫做方法

*## 具体的可到菜鸟教程上看 - - > ： https://www.runoob.com/php/php-oop.html*

* 了解了类和对象后，我们就要看看serialize函数的作用



结果为：O:4:"NISA":2:{s:4:"test";i:123;s:3:"ccc";s:3:"abc";}



我们对这个 ”压缩” 后的数据进行分析

O:4:"NISA":2:{s:4:"test";i:123;s:3:"ccc";s:3:"abc";}

O：表示object，表示这是一个对象；A为数组(array)；

4：表示对象名的长度，即“NISA”有四个字符长度

“NISA”:对象名

2：对象中有两个属性，即$test 和 $ccc

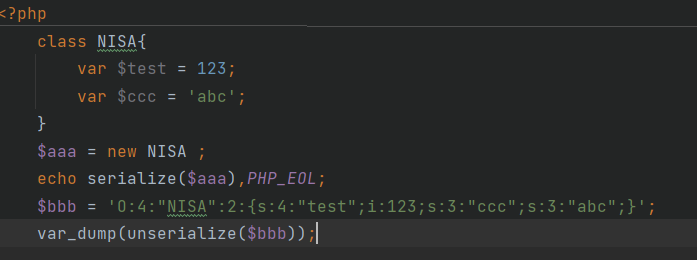
s: 表示第一个属性为string型

4：属性名的长度

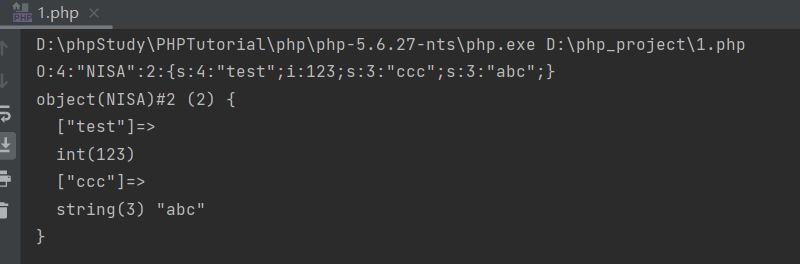
“test”：属性名

i：属性所对应的值的类型，比如后面的abc对应的s

我们再将这串字符反序列化



可以发现得到了原来的数据



以上就是对序列化和反序列化的简单介绍

* 补充

类中还有public、private、protect作为变量的调用权力

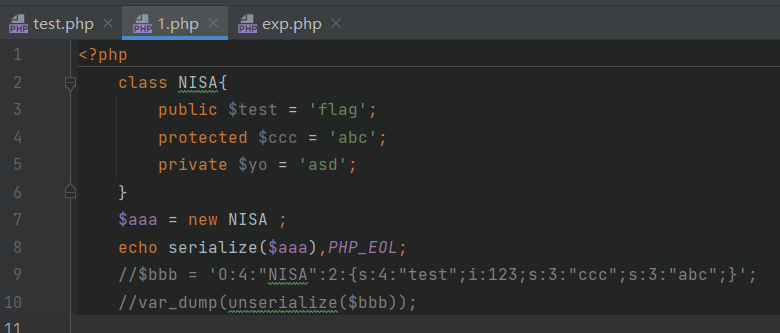
Public：任意地方都可以调用

Protect：只能在父类或者其继承子类中调用

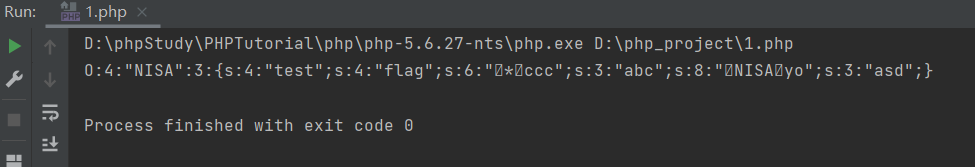
Private：只能在自己的类中调用

对于加了这些关键字的属性，序列化后会有不一样的效果

代码



输出结果



可以看到public的属性对应的没有改变

Protect对应的属性名加上了 %00\*%00 ， （%00是空白符）

Private对应的属性名加上了 %00对象名%00

**二、魔术方法**

其实反序列化的本身并没有什么危害，真正造成反序列化构成漏洞的还是反序列化中的魔术方法，接下来介绍反序列化当中一些常见的魔术方法

\_\_wakeup() //使用unserialize时触发

\_\_sleep() //使用serialize时触发

\_\_construct //对象被创建时触发

\_\_destruct() //对象被销毁时触发

\_\_call() //在对象上下文中调用不可访问的方法时触发

\_\_callStatic() //在静态上下文中调用不可访问的方法时触发

\_\_get() //用于从不可访问的属性读取数据

\_\_set() //用于将数据写入不可访问的属性

\_\_isset() //在不可访问的属性上调用isset()或empty()触发

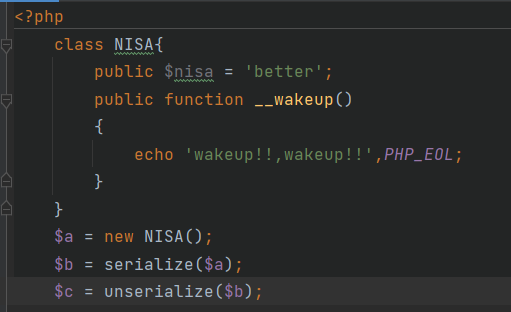
\_\_unset() //在不可访问的属性上使用unset()时触发

\_\_toString() //把类当作字符串使用时触发

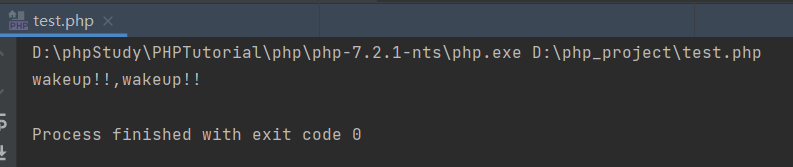
\_\_invoke() //当脚本尝试将对象调用为函数时触发

接下来将以echo来演示一些魔术方法

**·\_\_wakeup**



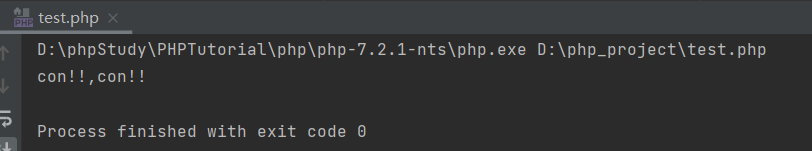
将会输出



**·\_\_construct**

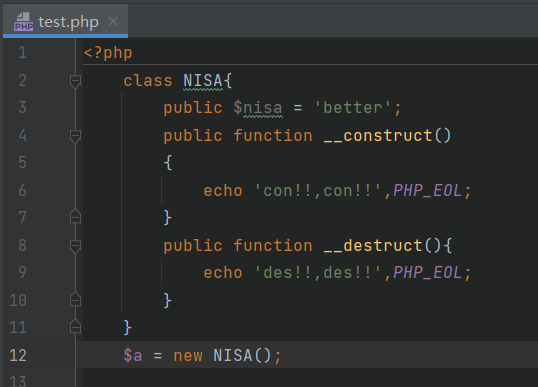


将会输出

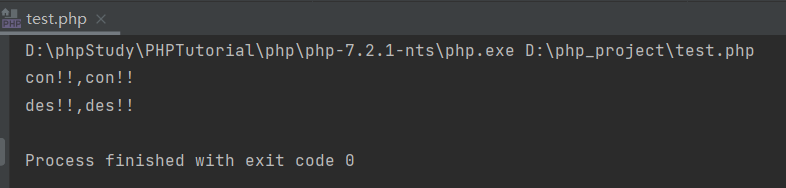


**·\_\_destruct**

当一段程序运行完时，对象会被自动销毁，就会触发\_\_destruct魔术方法

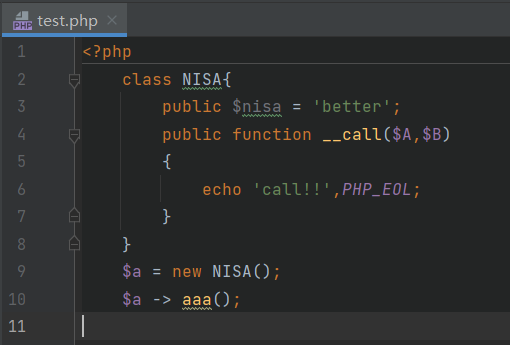


输出结果：

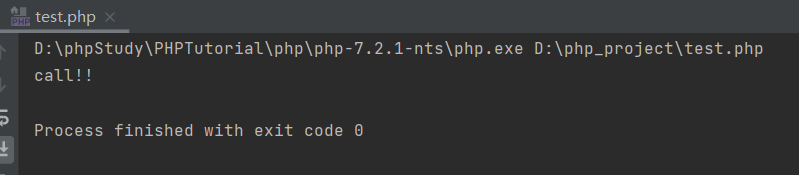


**·\_\_call**

前文有说，类里的函数称为方法，所以只要访问类里不存在的方法就能触发



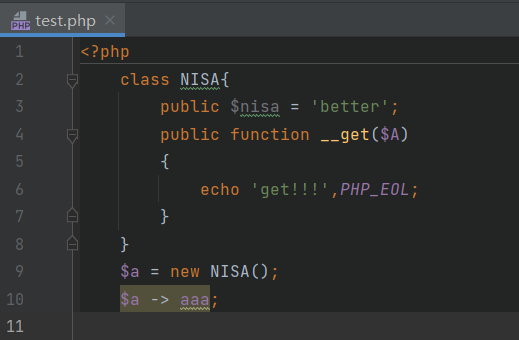
输出为



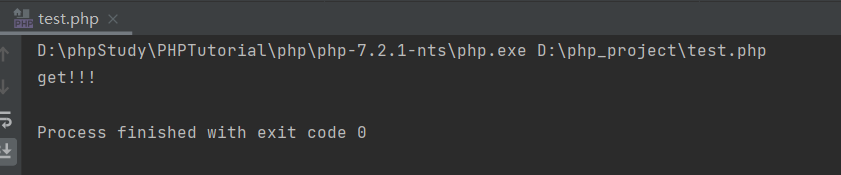
*## \_\_call后面要加两个参数；aaa()为NISA类中不存在的方法，所以调用\_\_call方法*

**·\_\_get**

类中的变量称为属性，访问类中不存在的属性时会触发\_\_get方法



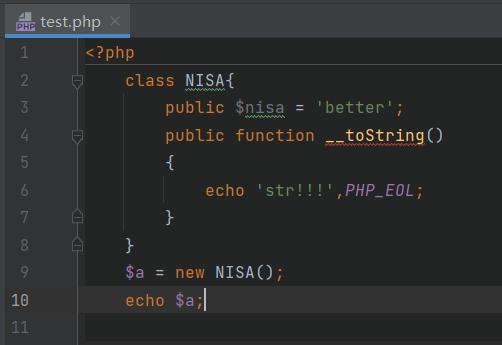
输出结果



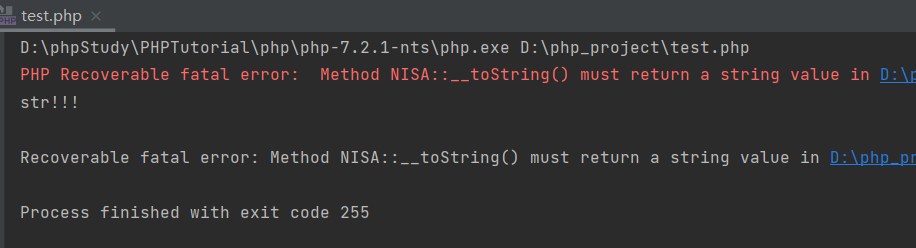
*## aaa为类中不存在的属性（存在的名为nisa），所以触发\_\_get方法(要有一个参数)*

**·\_\_toString**

输出对象不能用echo，echo是将变量作为字符串输出，所以用echo可以触发类中的\_\_toString方法。正确输出类要用var\_dump



输出结果

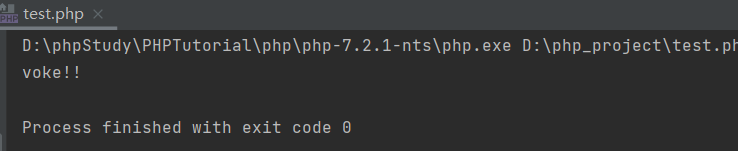


**·\_\_invoke**

在程序中，一般后面加上括号，就被判定为函数调用，用这个来触发\_\_invoke方法



输出结果为



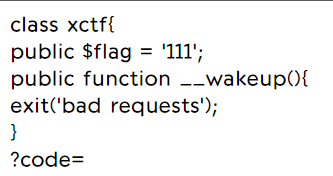
* 通常题目都会有多个魔术方法、多个类，类之间会互相调用，我们需要根据类之间的调用关系来构造题目所需的pop链，像eval、include等都为敏感函数，在写pop链时，应以这些敏感函数作为最终目的去构造。

**三、PHP反序列化考点**

**①\_\_wakeup绕过**

当你的反序列化字符串中属性个数大于实际属性个数时，会绕过\_\_wakeup的判断

攻防世界：<https://adworld.xctf.org.cn/task/answer?type=web&number=3&grade=1&id=4821&page=1>



这题正常情况下的payload应为O:4:"xctf":1:{s:4:"flag";s:3:"111";}

而为了绕过\_\_wakeup，我们需要将属性个数的1改为2

O:4:"xctf":2:{s:4:"flag";s:3:"111";} 完成绕过

**②Bypass反序列化正则绕过**

题目有时会对反序列化的值进行正则拦截，正则式为'/[oc]:\d+:/i' ，也就是过滤了’O:数字’

我们用加号进行绕过，O:4 等价于O:+4 ，完成绕过

- - 题目练习 -- > 攻防世界 ：

https://adworld.xctf.org.cn/task/answer?type=web&number=3&grade=1&id=5409&page=1

**③PHP原生类的利用**

PHP中有自己内置的原生类，里边也有一些魔术方法可以供我们调用

常用的原生类有

* Error
* Exception
* SoapClient
* DirectoryIterator
* SimpleXMLElement

1. Exception类（\_\_tostring方法）

Php5、7有

<?php

$a = new Exception("<script>alert(1)</script>");

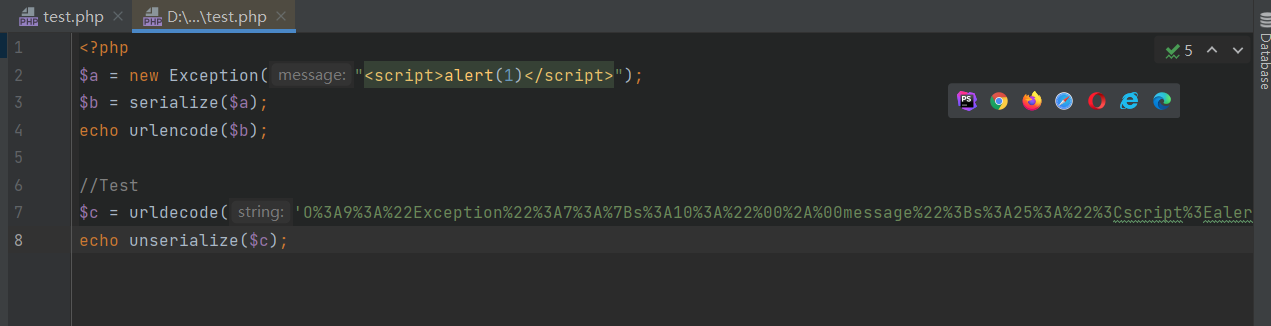
$b = serialize($a);

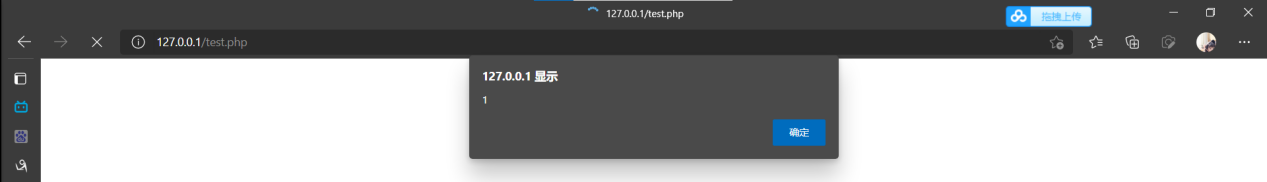
echo urlencode($b);

$c = urldecode('O%3A9%3A%22Exception%22%3A7%3A%7Bs%3A10%3A%22%00%2A%00message%22%3Bs%3A25%3A%22%3Cscript%3Ealert%281%29%3C%2Fscript%3E%22%3Bs%3A17%3A%22%00Exception%00string%22%3Bs%3A0%3A%22%22%3Bs%3A7%3A%22%00%2A%00code%22%3Bi%3A0%3Bs%3A7%3A%22%00%2A%00file%22%3Bs%3A18%3A%22%2Fusercode%2Ffile.php%22%3Bs%3A7%3A%22%00%2A%00line%22%3Bi%3A2%3Bs%3A16%3A%22%00Exception%00trace%22%3Ba%3A0%3A%7B%7Ds%3A19%3A%22%00Exception%00previous%22%3BN%3B%7D');

echo unserialize($c);

可实现XSS





1. Error类（\_\_tostring方法）

Php7有，同Exception类一样也是实现XSS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* 使用 Error/Exception 内置类绕过哈希比较 \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

我们先看一下两个类在内置的定义

**Error类：**

Error implements Throwable {

/\* 属性 \*/

protected string $message ;

protected int $code ;

protected string $file ;

protected int $line ;

/\* 方法 \*/

public \_\_construct ( string $message = "" , int $code = 0 , Throwable $previous = null )

final public getMessage ( ) : string

final public getPrevious ( ) : Throwable

final public getCode ( ) : mixed

final public getFile ( ) : string

final public getLine ( ) : int

final public getTrace ( ) : array

final public getTraceAsString ( ) : string

public \_\_toString ( ) : string

final private \_\_clone ( ) : void

}

**Exception类：**

Exception {

/\* 属性 \*/

protected string $message ;

protected int $code ;

protected string $file ;

protected int $line ;

/\* 方法 \*/

public \_\_construct ( string $message = "" , int $code = 0 , Throwable $previous = null )

final public getMessage ( ) : string

final public getPrevious ( ) : Throwable

final public getCode ( ) : mixed

final public getFile ( ) : string

final public getLine ( ) : int

final public getTrace ( ) : array

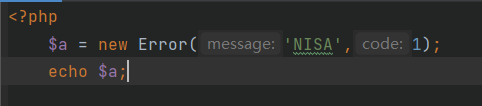
final public getTraceAsString ( ) : string

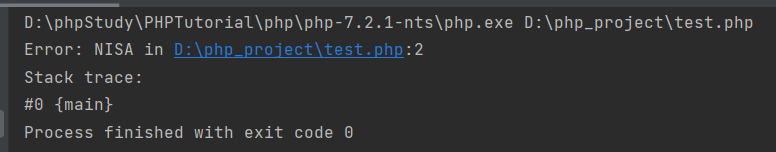
public \_\_toString ( ) : string

final private \_\_clone ( ) : void

}

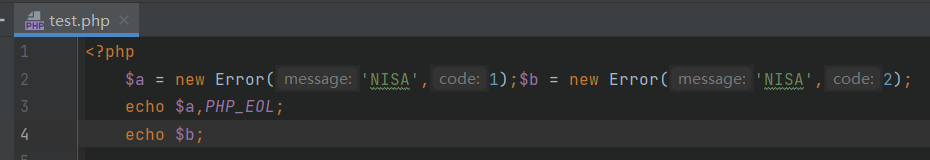
我们以error类为例，看看触发\_\_toString方法是什么样子

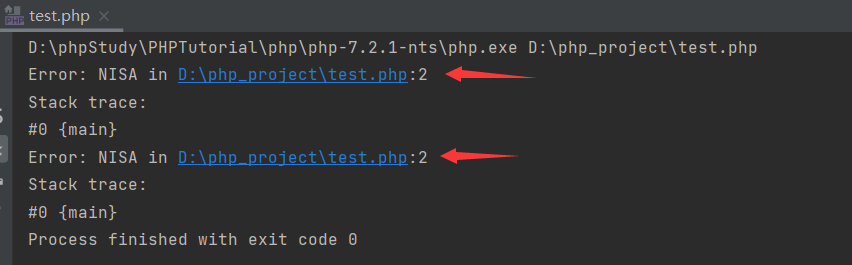




可以发现我们的NISA被输出出来了，但是我们后面的1没有被显示出来

我们再来看下一个例子：





可以发现，我们虽然构造了两个不同信息的类（一个code为1，一个为2），但是输出结果是一致的。

*## 两个变量必须放在同一行，因为输出结果中会带有行号*

利用这一点，我们可以绕过php类中的哈希比较

**[2020 极客大挑战]Greatphp**

给出源码：

<?php

error\_reporting(0);

class SYCLOVER {

public $syc;

public $lover;

public function \_\_wakeup(){

if( ($this->syc != $this->lover) && (md5($this->syc) === md5($this->lover)) && (sha1($this->syc)=== sha1($this->lover)) ){

if(!preg\_match("/\<\?php|\(|\)|\"|\'/", $this->syc, $match)){

eval($this->syc);

} else {

die("Try Hard !!");

}

}

}

}

if (isset($\_GET['great'])){

unserialize($\_GET['great']);

} else {

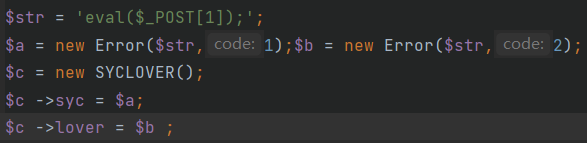
highlight\_file(\_\_FILE\_\_);

}

?>

很显然，我们要想走到eval函数，必须要绕过判断，通常ctf中的哈希比较，我们都用数组来绕过，类中的哈希比较还有一种思路就是通过两个内置类来绕过

假设这题需要的eval的内容为eval($\_POST[1]) , 那我们可以构造成这样：



从而绕过类中的哈希比较

*## 这题因为比较完还有正则过滤，所以假设中的payload是不行的，这里只是举个栗子*

1. SimpleXMLElement类（\_\_tostring方法）

实现XXE

1. SoapClient类（\_\_call方法）

配合SSRF

这两类涉及的XXE和SSRF，我还没学到，就先不展开介绍

参考文章：https://www.anquanke.com/post/id/238482

**④phar反序列化**

1. phar简介

简单来说phar就是php压缩文档。它可以把多个文件归档到同一个文件中，而且不经过解压就能被 php 访问并执行，与file:// php://等类似，也是一种流包装器。

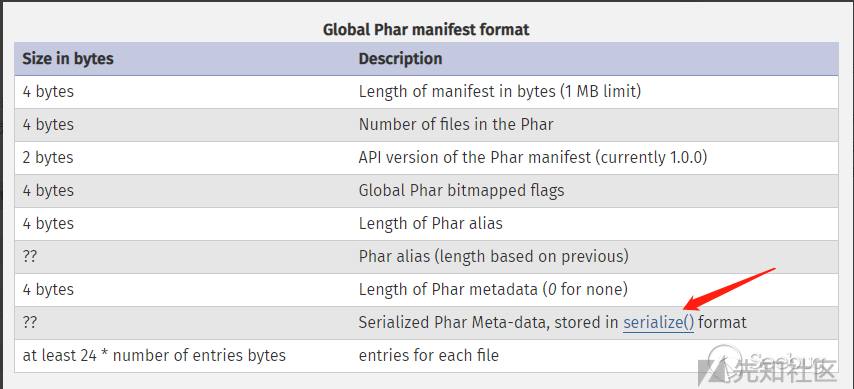
1. phar文件结构

1. stub

stub的基本结构：xxx<?php xxx;\_\_HALT\_COMPILER();?>，前面内容不限，但必须以\_\_HALT\_COMPILER();?>来结尾，否则phar扩展将无法识别这个文件为phar文件。

2.manifest

Phar文件中被压缩的文件的一些信息，其中Meta-data部分的信息会以序列化的形式储存，这里就是漏洞利用的关键点



3.contents

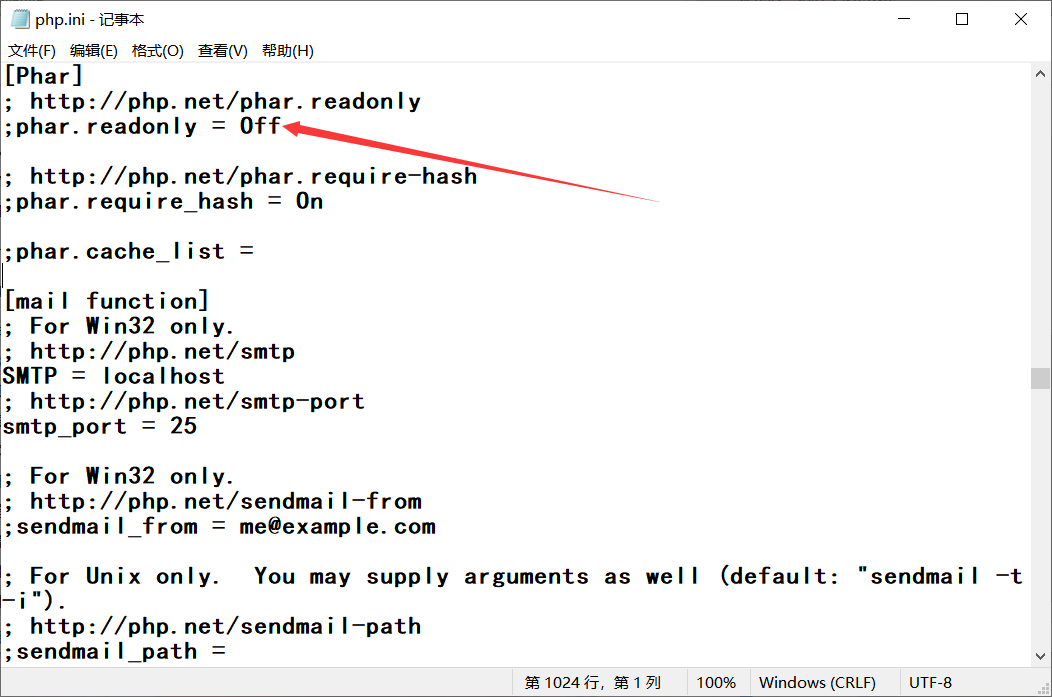
被压缩的文件内容，在没有特殊要求的情况下，这个被压缩的文件内容可以随便写的，因为我们利用这个漏洞主要是为了触发它的反序列化

1. signature

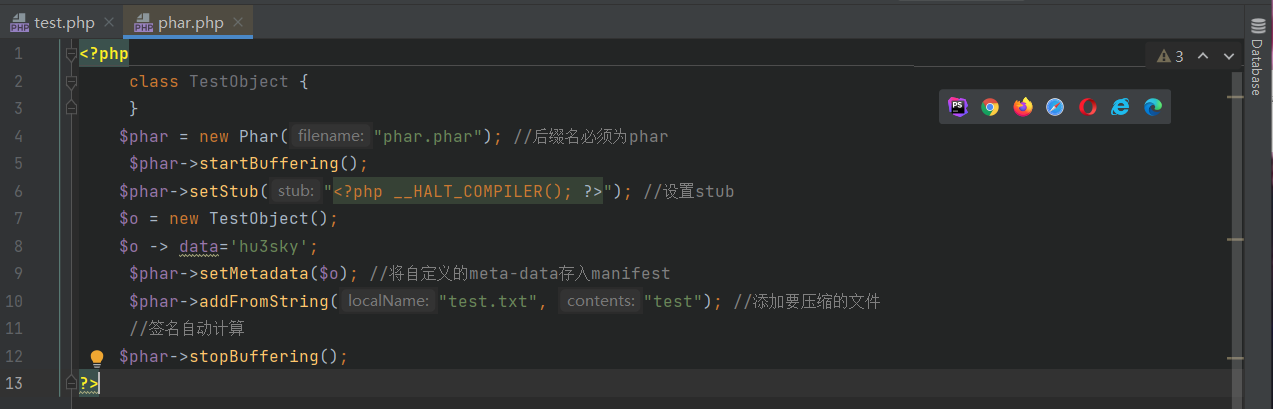
签名，放在文件末尾

* 示例↓：

*# 先将此处参数设置为off*



写phar.php



 <?php

class TestObject {

}

$phar = new Phar("phar.phar"); //后缀名必须为phar

$phar->startBuffering();

$phar->setStub("<?php \_\_HALT\_COMPILER(); ?>"); //设置stub

$o = new TestObject();

$o -> data='hu3sky';

$phar->setMetadata($o); //将自定义的meta-data存入manifest

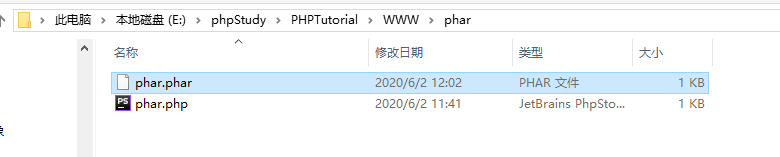
$phar->addFromString("test.txt", "test"); //添加要压缩的文件

//签名自动计算

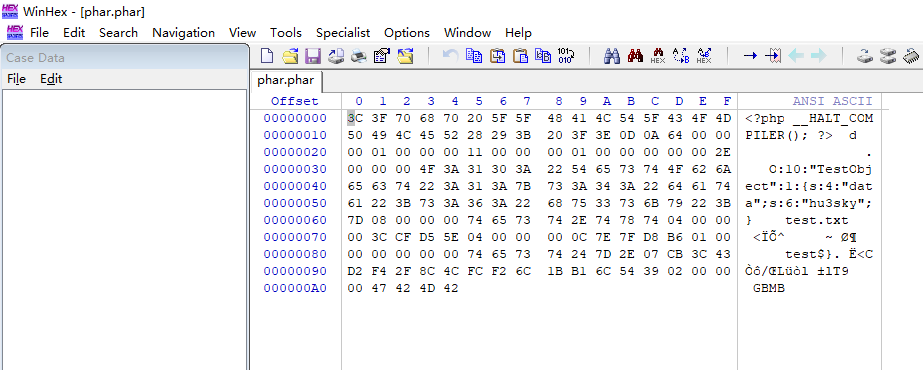
$phar->stopBuffering();

?>

本地访问这个phar.php，会自动生成一个phar.phar



再放在winhex下查看，可以看到我们写入的数据被序列化存储



再创建一个phar\_fan.php

<?php

class TestObject{

function \_\_destruct()

{

echo $this -> data;

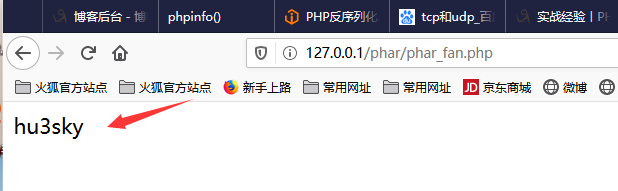
}

}

include('phar://phar.phar');

?>

访问后成功触发反序列化



我们可以对反序列化的内容进行控制，从而达到我们想要的效果

**⑤其他**

其他的还有反序列化字符逃逸、session反序列化等，自行查阅