php反序列化总结——张晏龄

目录

**[PHP反序列化 2](#_Toc23894)**

[一．简介 2](#_Toc16146)

[案例引入 2](#_Toc11976)

[二．魔术方法 4](#_Toc599)

[三．对象注入 10](#_Toc28435)

[四．POP链的构造利用 10](#_Toc5718)

[五．PHP原生类反序列化利用 12](#_Toc32234)

[六．Phar反序列化 14](#_Toc25693)

[七．php-session反序列化 18](#_Toc7009)

# PHP反序列化

## 一．简介

序列化其实就是将数据转化成一种可逆的数据结构，自然，逆向的过程就叫做反序列化。在网上找到一个比较形象的例子:

**比如**：现在我们都会在淘宝上买桌子，桌子这种很不规则的东西，该怎么从一个城市运输到另一个城市，这时候一般都会把它拆掉成板子，再装到箱子里面，就可以快递寄出去了，这个过程就类似我们的序列化的过程（把数据转化为可以存储或者传输的形式）。当买家收到货后，就需要自己把这些板子组装成桌子的样子，这个过程就像反序列的过程（转化成当初的数据对象）。

php 将数据序列化和反序列化会用到两个函数:

serialize :将对象格式化成有序的字符串

unserialize :将字符串还原成原来的对象

序列化的目的是方便数据的传输和存储，在PHP中，序列化和反序列化一般用做缓存，比如session缓存，cookie等。

常见的序列化格式：

1.二进制格式

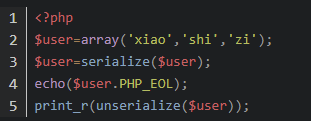
2.字节数组

3.json字符串

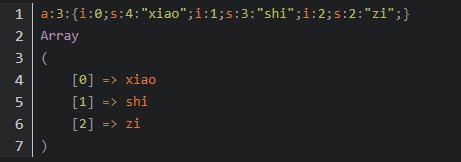
4.xml字符串

## **案例引入**

* 简单的例子（以数组为例子）



输出：



a:3:{i:0;s:4:"xiao";i:1;s:3:"shi";i:2;s:2:"zi";}

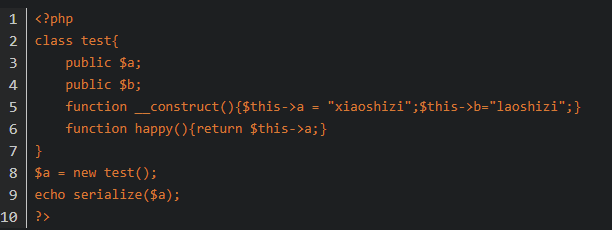
a: array代表是数组，后面的3说明有三个属性

i: 代表是整型数据int，后面的0是数组下标

s: 代表是字符串，后面的4是因为xiao长度为4

依次类推

* 序列化后的内容只有成员变量，没有成员函数，比如下面的例子：



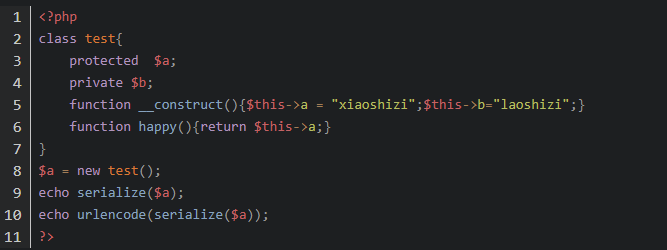
输出(O代表Object是对象的意思，也是类)：



变量前是protected，会在变量名前加上\x00\*\x00,private

变量前是private，会在变量名前加上\x00类名\x00

输出时一般需要url编码，若在本地存储更推荐采用base64编码的形式，如下：



输出则会导致不可见字符\x00的丢失：



## 二．魔术方法

\_\_wakeup() //执行unserialize()时，先会调用这个函数

\_\_sleep() //执行serialize()时，先会调用这个函数

\_\_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或者不存在这个键都会调用此方法

\_\_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

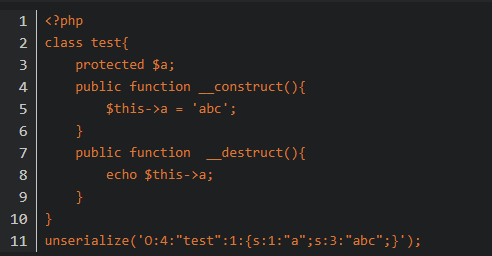
\_\_invoke() //当尝试将对象调用为函数时触发

## 反序列化绕过小Trick：

### **1.php7.1+反序列化对类属性不敏感**

如果变量前是protected，序列化结果会在变量名前加上\x00\*\x00

但在特定版本7.1以上则对于类属性不敏感，比如下面的例子即使没有\x00\*\x00也依然会输出abc：



### **2.绕过\_\_wakeup**

版本：

PHP5 < 5.6.25

PHP7 < 7.0.10

**利用方式**：序列化字符串中表示对象属性个数的值大于真实的属性个数时会跳过\_\_wakeup的执行

对于下面这样一个自定义类：



如果执行unserialize('O:4:"test":1:{s:1:"a";s:3:"abc";}');输出结果为666

而把对象属性个数的值增大执行unserialize('O:4:"test":2:{s:1:"a";s:3:"abc";}');输出结果为abc

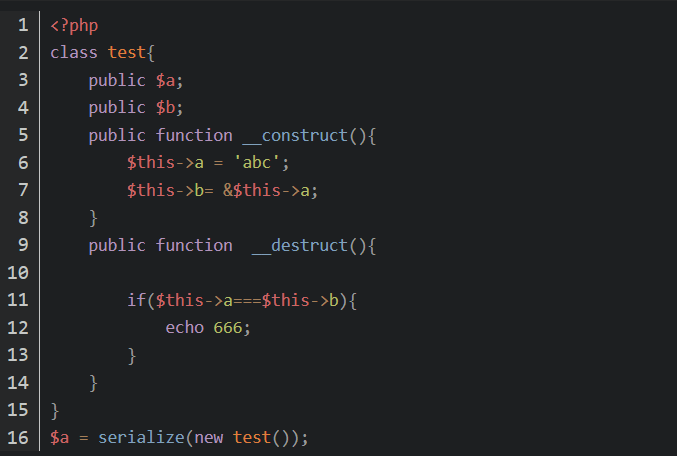
**3.绕过部分正则**

preg\_match('/^O:\d+/')匹配序列化字符串是否是对象字符串开头,这在曾经的CTF中也出过类似的考点

* 利用加号绕过（注意在url里传参时+要编码为%2B）
* serialize(array(a ) ) ; / / a));//a));//a为要反序列化的对象(序列化结果开头是a，不影响作为数组元素的$a的析构)

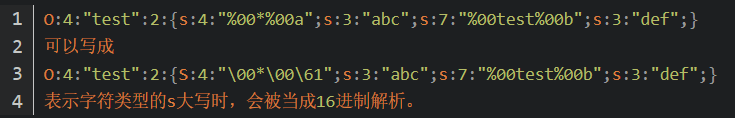


### **4.利用引用**



上面这个例子将$b设置为$a的引用，可以使$a永远与$b相等

### **16进制绕过字符的过滤**



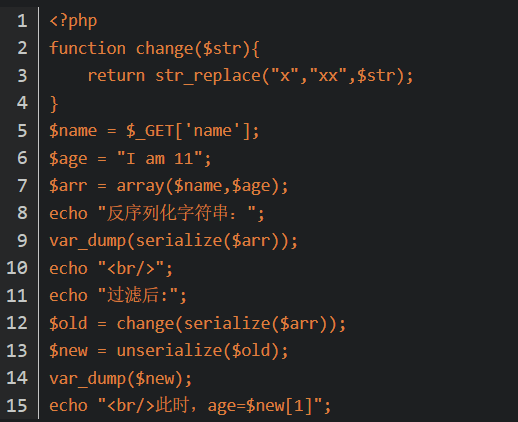
### 表示字符类型的S大写时，会被当作16进制解析



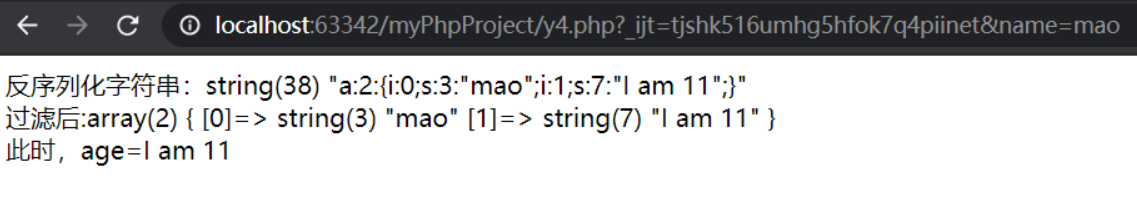
### **PHP反序列化字符逃逸**

#### 情况1：过滤后字符变多

首先给出本地的php代码，把反序列化后的一个x替换成为两个

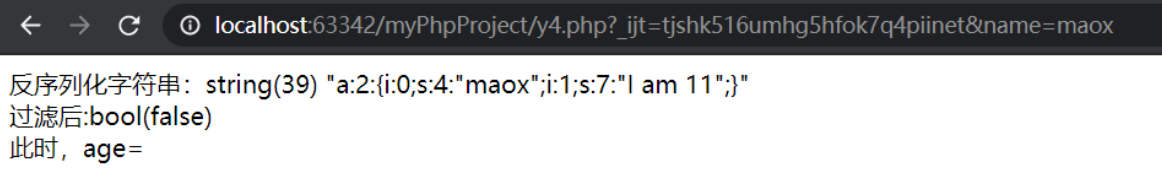


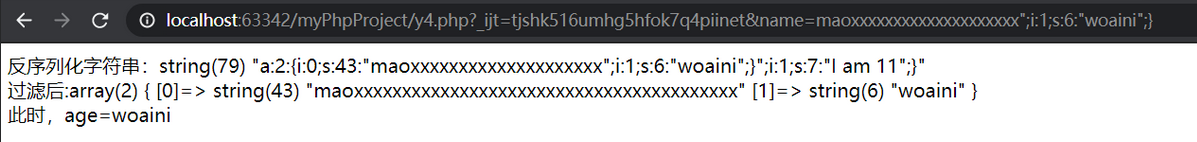
正常情况,传入name=mao



如果此时多传入一个x的话会怎样，毫无疑问反序列化失败，由于溢出(s本来是4结果多了一个字符出来)，我们可以利用这一点实现字符串逃逸

令





我们传入name=maoxxxxxxxxxxxxxxxxxxxx";i:1;s:6:"woaini";}

";i:1;s:6:"woaini";}这一部分一共20个字符

由于一个x会被替换为两个，我们输入了一共20个x，现在是40个，多出来的20个x其实取代了我们的这二十个字符";i:1;s:6:"woaini";}，从而造成";i:1;s:6:"woaini";}的溢出

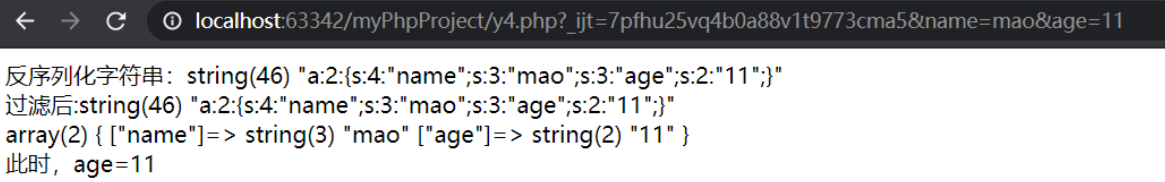
* 前面的"闭合了前串，使得我们的字符串成功逃逸，可以被反序列化，输出woaini
* 最后的;}闭合反序列化全过程导致原来的";i:1;s:7:"I am 11";}"被舍弃，不影响反序列化过程

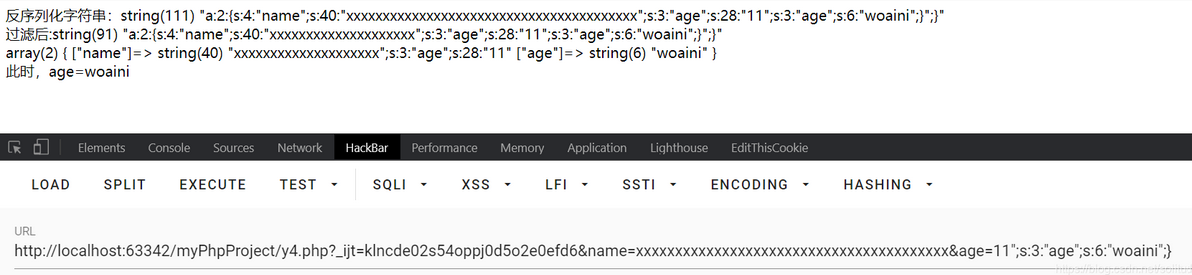
#### 情况2：过滤后字符变少

把反序列化后的两个x替换成为一个



正常情况传入name=mao&age=11的结果





前面少了一半，导致后面的字符被吃掉，从而执行了我们后面的代码；  
这部分是age序列化后的结果：

s:3:"age";s:28:"11";s:3:"age";s:6:"woaini";}"

前面是40个x所以导致少了20个字符，所以需要后面来补上，";s:3:"age";s:28:"11这一部分刚好20个

后面有"闭合了前面因此后面的参数就可以由我们自定义执行了

## 三．对象注入

当用户的请求在传给反序列化函数unserialize()之前没有被正确的过滤时就会产生漏洞。因为PHP允许对象序列化，攻击者就可以提交特定的序列化的字符串给一个具有该漏洞的unserialize函数，最终导致一个在该应用范围内的任意PHP对象注入。

**对象漏洞**出现得满足两个前提：

**1、unserialize的参数可控**。

**2、 代码里有定义一个含有魔术方法的类，并且该方法里出现一些使用类成员变量作为参数的存在安全问题的函数。**



在脚本运行结束后便会调用\_destruct函数，同时会覆盖test变量输出maomi

## 四．POP链的构造利用

当关键代码不在魔术方法中，而是在一个类的普通方法中。这时候可以通过寻找相同的函数名将类的属性和敏感函数的属性联系起来

* **例**MRCTF2020-Ezpop



**利用思路**：

首先逆向分析，我们最终是希望通过Modifier当中的append方法实现本地文件包含读取文件，回溯到调用它的\_\_invoke，当我们将对象调用为函数时触发,发现在Test类当中的\_\_get方法，再回溯到Show当中的\_\_toString，再回溯到Show当中的\_\_construct当中有echo $this->source可以调用\_\_toString

因此不难构造pop链：



## 五．PHP原生类反序列化利用

SoapClient介绍

php在安装php-soap拓展后，可以反序列化原生类SoapClient，来发送http post请求。

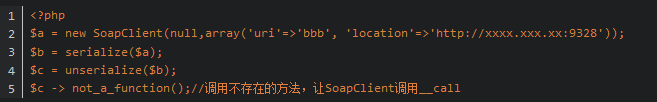
必须调用SoapClient不存在的方法，触发SoapClient的\_\_call魔术方法。

通过CRLF来添加请求体：SoapClient可以指定请求的user-agent头，通过添加换行符的形式来加入其他请求内容

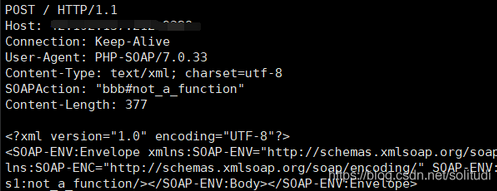
SoapClient采用了HTTP作为底层通讯协议，XML作为数据传送的格式，其采用了SOAP协议(SOAP 是一种简单的基于 XML 的协议,它使应用程序通过 HTTP 来交换信息)，其次我们知道某个实例化的类，如果去调用了一个不存在的函数，会去调用\_\_call方法

**利用方式**

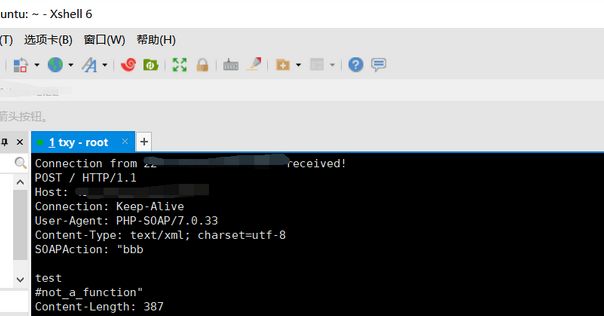
下面首先在我的VPS上面开启监听nc -lvvp 9328



运行上面的php程序，在我的vps上面奖会捕获监听

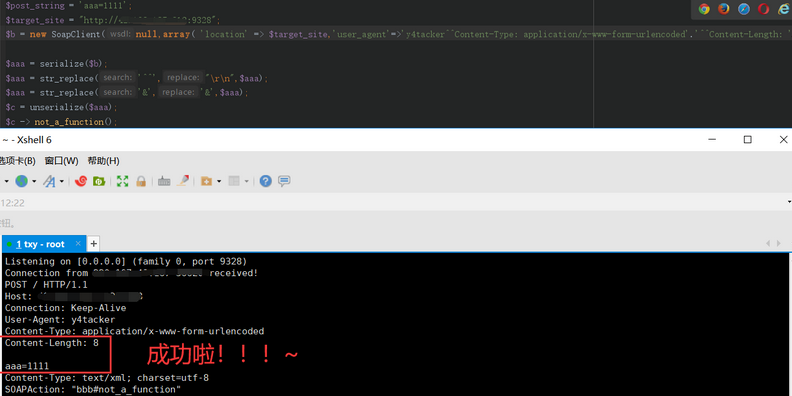


从上面这张图可以看到，SOAPAction处是我们的可控参数，因此我们可以尝试注入我们自己恶意构造的CRLF即插入\*\*\r\n\*\*，利用成功！

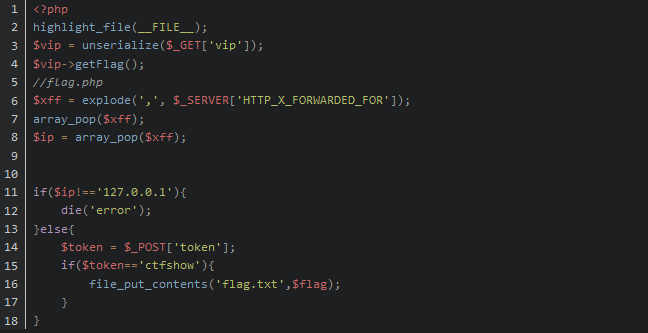


但是还有个问题，我们再发送POST数据的时候是需要遵循HTTP协议，指定请求头Content-Type: application/x-www-form-urlencoded但Content-Type在SOAPAction的上面，就无法控制Content-Type,也就不能控制POST的数据

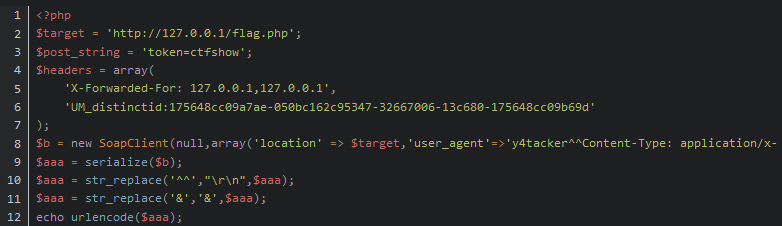
接下来我们实验一下



### **例：反序列化我们传入的vip执行getFlag函数(迷惑人的函数)**



由于服务器带有cloudfare代理，我们无法通过本地构造XFF头实现绕过，我们需要使用SoapClient与CRLF实现SSRF访问127.0.0.1/flag.php,即可绕过cloudfare代理



接下来访问flag.txt即可

## 六．Phar反序列化

phar文件本质上是一种压缩文件，会以序列化的形式存储用户自定义的meta-data。当受影响的文件操作函数调用phar文件时，会自动反序列化meta-data内的内容。

**1.phar文件：**

在软件中，PHAR（PHP归档）文件是一种打包格式，通过将许多PHP代码文件和其他资源（例如图像，样式表等）捆绑到一个归档文件中来实现应用程序和库的分发

php通过用户定义和内置的“流包装器”实现复杂的文件处理功能。内置包装器可用于文件系统函数，如(fopen(),copy(),file\_exists()和filesize()。 phar://就是一种内置的流包装器。

**2.php中一些常见的流包装器如下：**

file:// — 访问本地文件系统，在用文件系统函数时默认就使用该包装器

http:// — 访问 HTTP(s) 网址

ftp:// — 访问 FTP(s) URLs

php:// — 访问各个输入/输出流（I/O streams）

zlib:// — 压缩流

data:// — 数据（RFC 2397）

glob:// — 查找匹配的文件路径模式

phar:// — PHP 归档

ssh2:// — Secure Shell 2

rar:// — RAR

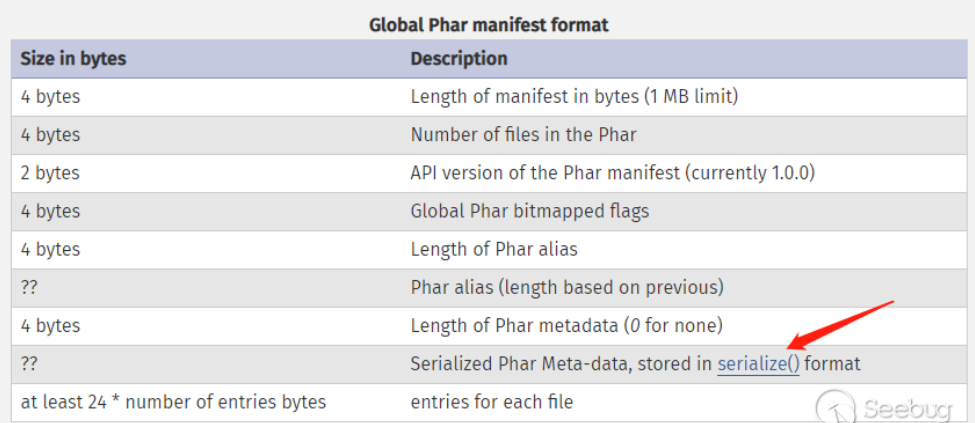
ogg:// — 音频流

expect:// — 处理交互式的流

### **3.phar文件的结构**

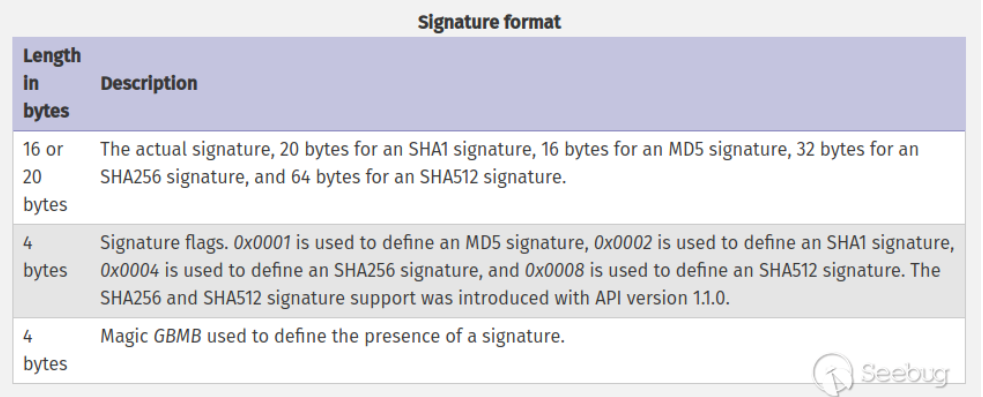
stub:phar文件的标志，格式为xxx<?php xxx; \_\_HALT\_COMPILER();?>，前面内容不限，必须以 xxx \_\_HALT\_COMPILER();?> 结尾，否则无法识别。xxx可以为自定义内容。

manifest:phar文件本质上是一种压缩文件，其中每个被压缩文件的权限、属性等信息都放在这部分。这部分还会以序列化的形式存储用户自定义的meta-data，这是漏洞利用最核心的地方。



content:被压缩文件的内容

signature (可空):签名，放在末尾。



**4.如何生成一个phar文件？**

* 例：

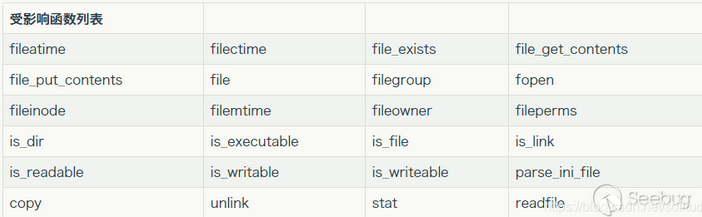


### **5.漏洞利用条件**

1. phar文件要能够上传到服务器端。
2. 要有可用的魔术方法作为“跳板”。
3. 文件操作函数的参数可控，且:、/、phar等特殊字符没有被过滤

### **6.受影响的函数**

知道创宇测试后受影响的函数列表：



还有：

//exif

exif\_thumbnail

exif\_imagetype

//gd

imageloadfont

imagecreatefrom\*\*\*系列函数

//hash

hash\_hmac\_file

hash\_file

hash\_update\_file

md5\_file

sha1\_file

// file/url

get\_meta\_tags

get\_headers

//standard

getimagesize

getimagesizefromstring

// zip

$zip = new ZipArchive();

$res = $zip->open('c.zip');

$zip->extractTo('phar://test.phar/test');

// Bzip / Gzip 当环境限制了phar不能出现在前面的字符里。可以使用compress.bzip2://和compress.zlib://绕过

$z = 'compress.bzip2://phar:///home/sx/test.phar/test.txt';

$z = 'compress.zlib://phar:///home/sx/test.phar/test.txt';

//配合其他协议：(SUCTF)

//https://www.xctf.org.cn/library/details/17e9b70557d94b168c3e5d1e7d4ce78f475de26d/

//当环境限制了phar不能出现在前面的字符里，还可以配合其他协议进行利用。

//php://filter/read=convert.base64-encode/resource=phar://phar.phar

//Postgres pgsqlCopyToFile和pg\_trace同样也是能使用的，需要开启phar的写功能。

<?php

$pdo = new PDO(sprintf("pgsql:host=%s;dbname=%s;user=%s;password=%s", "127.0.0.1", "postgres", "sx", "123456"));

@$pdo->pgsqlCopyFromFile('aa', 'phar://phar.phar/aa');

?>

// Mysql

//LOAD DATA LOCAL INFILE也会触发这个php\_stream\_open\_wrapper

//配置一下mysqld:

//[mysqld]

//local-infile=1

//secure\_file\_priv=""

<?php

class A {

public $s = '';

public function \_\_wakeup () {

system($this->s);

}

}

$m = mysqli\_init();

mysqli\_options($m, MYSQLI\_OPT\_LOCAL\_INFILE, true);

$s = mysqli\_real\_connect($m, 'localhost', 'root', 'root', 'testtable', 3306);

$p = mysqli\_query($m, 'LOAD DATA LOCAL INFILE \'phar://test.phar/test\' INTO TABLE a LINES TERMINATED BY \'\r\n\' IGNORE 1 LINES;');

?>

### **7.绕过方式**

当环境限制了phar不能出现在前面的字符里。可以使用compress.bzip2://和compress.zlib://等绕过

compress.bzip://phar:///test.phar/test.txt

compress.bzip2://phar:///test.phar/test.txt

compress.zlib://phar:///home/sx/test.phar/test.txt

php://filter/resource=phar:///test.phar/test.txt

### 当环境限制了phar不能出现在前面的字符里，还可以配合其他协议进行利用。

php://filter/read=convert.base64-encode/resource=phar://phar.phar

GIF格式验证可以通过在文件头部添加GIF89a绕过

1、$phar->setStub(“GIF89a”."<?php \_\_HALT\_COMPILER(); ?>"); //设置stub

2、生成一个phar.phar，修改后缀名为phar.gif

## 七．php-session反序列化

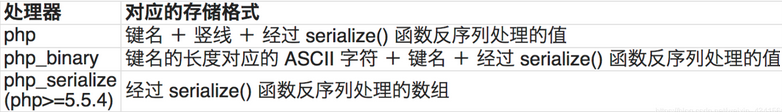
### **1.session**

在计算机中，尤其是在网络应用中，称为“会话控制”。Session 对象存储特定用户会话所需的属性及配置信息。这样，当用户在应用程序的 Web 页之间跳转时，存储在 Session 对象中的变量将不会丢失，而是在整个用户会话中一直存在下去。当用户请求来自应用程序的 Web 页时，如果该用户还没有会话，则 Web 服务器将自动创建一个 Session 对象。当会话过期或被放弃后，服务器将终止该会话。

当第一次访问网站时，Seesion\_start()函数就会创建一个唯一的Session ID，并自动通过HTTP的响应头，将这个Session ID保存到客户端Cookie中。同时，也在服务器端创建一个以Session ID命名的文件，用于保存这个用户的会话信息。当同一个用户再次访问这个网站时，也会自动通过HTTP的请求头将Cookie中保存的Seesion ID再携带过来，这时Session\_start()函数就不会再去分配一个新的Session ID，而是在服务器的硬盘中去寻找和这个Session ID同名的Session文件，将这之前为这个用户保存的会话信息读出，在当前脚本中应用，达到跟踪这个用户的目的。

### **2.session 的存储机制**

php中的session中的内容并不是放在内存中的，而是以文件的方式来存储的，存储方式就是由配置项session.save\_handler来进行确定的，默认是以文件的方式存储。  
存储的文件是以sess\_sessionid来进行命名的



### **3.php.ini中一些session配置**

session.save\_path="" --设置session的存储路径

session.save\_handler=""–设定用户自定义存储函数，如果想使用PHP内置会话存储机制之外的可以使用本函数(数据库等方式)

session.auto\_start boolen–指定会话模块是否在请求开始时启动一个会话默认为0不启动

session.serialize\_handler string–定义用来序列化/反序列化的处理器名字。默认使用php

**4.利用姿势**

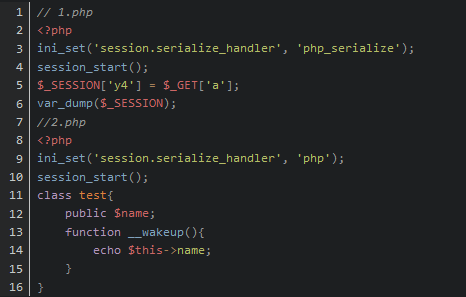
session.upload\_progress进行文件包含和反序列化渗透

使用不同的引擎来处理session文件

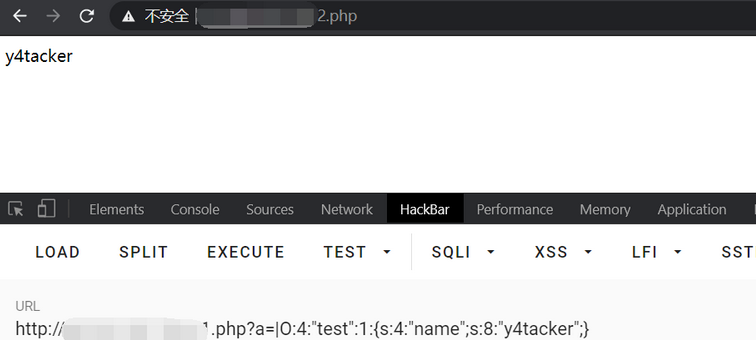
1.$\_SESSION变量直接可控

php引擎的存储格式是键名|serialized\_string，而php\_serialize引擎的存储格式是serialized\_string。如果程序使用两个引擎来分别处理的话就会出现问题

来看看这两个php



首先访问1.php，传入参数a=|O:4:"test":1:{s:4:"name";s:8:"y4tacker";}再访问2.php，注意不要忘记|



由于1.php是使用php\_serialize引擎处理，因此只会把'|'当做一个正常的字符。

然后访问2.php，由于用的是php引擎，因此遇到'|'时会将之看做键名与值的分割符，从而造成了歧义，导致其在解析session文件时直接对'|'后的值进行反序列化处理。

session\_start()这个函数官方说明：

当会话自动开始或者通过 session\_start() 手动开始的时候， PHP 内部会调用会话管理器的 open 和 read 回调函数。 会话管理器可能是 PHP 默认的， 也可能是扩展提供的（SQLite 或者 Memcached 扩展）， 也可能是通过 session\_set\_save\_handler() 设定的用户自定义会话管理器。 通过 read 回调函数返回的现有会话数据（使用特殊的序列化格式存储），PHP 会自动反序列化数据并且填充 $\_SESSION 超级全局变量

因此我们成功触发了test类中的\_\_wakeup()方法,所以这种攻击思路是可行的。但这种方法是在可以对session的进行赋值的

##### 2.$\_SESSION变量直接不可控

* 例：高校战疫的一道CTF题目



我们注意到这样一句话ini\_set('session.serialize\_handler', 'php');，因此不难猜测本身在php.ini当中的设置可能是php\_serialize，在查看了phpinfo后得证猜测正确，也知道了这道题的考点

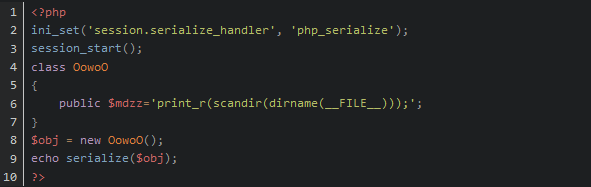
那么我们就进入phpinfo查看一下，enabled=on表示upload\_progress功能开始，也意味着当浏览器向服务器上传一个文件时，php将会把此次文件上传的详细信息(如上传时间、上传进度等)存储在session当中 ；只需往该地址任意 POST 一个名为 PHP\_SESSION\_UPLOAD\_PROGRESS 的字段，就可以将filename的值赋值到session中



构造文件上传的表单

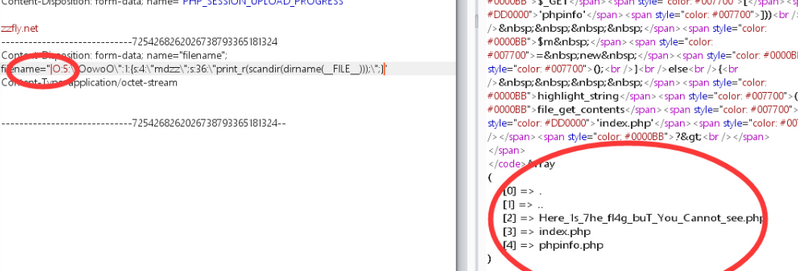


接下来构造序列化payload

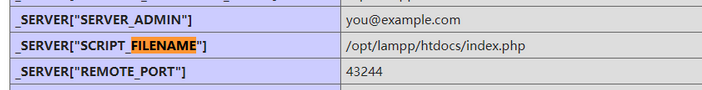


由于采用Burp发包，为防止双引号被转义，在双引号前加上\，除此之外还要加上|

在这个页面随便上传一个文件，然后抓包修改filename的值

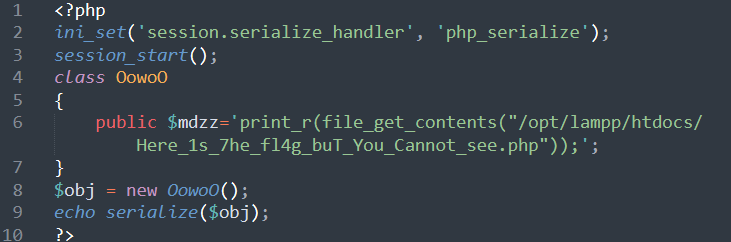


可以看到Here\_1s\_7he\_fl4g\_buT\_You\_Cannot\_see.php这个文件，flag肯定在里面，但还有一个问题就是不知道这个路径，路径的问题就需要回到phpinfo页面去查看



因此我们只需要把payload，当中改为

print\_r(file\_get\_contents("/opt/lampp/htdocs/Here\_1s\_7he\_fl4g\_buT\_You\_Cannot\_see.php"));即可获取flag



参考文章：

https://waterdrop.yuque.com/smq4eh/pm1lp9/qwxtti

https://blog.csdn.net/solitudi/article/details/113588692

https://www.freebuf.com/vuls/202819.html

https://blog.zsxsoft.com/post/38