

thinkphp3.2\_find\_select\_delete注入

[phpoop](#) / 2018-08-24 19:04:35 / 浏览数 4425 [安全技术](#) [漏洞分析](#) [顶\(1\)](#) [踩\(0\)](#)

推荐水泡师傅的分析感觉比我清楚：<https://xz.aliyun.com/t/2629>

## 0x00 简介

已下是阅读须知

本文所用框架是官方Thinkphp3.2.3

好没了，又不是BB机，没有那么多的骚话==

## 0x01 下载安装测试代码

下载地址：<http://www.thinkphp.cn/download/610.html>

自己配置一下数据库路径：test\_thinkphp\_3.2.3\Application\Common\Conf\config.php

```
index.php  config.php x
1  <?php
2  return array(
3      /**配置项'=>'配置值*/
4      'DB_TYPE'          => 'mysql',      // 数据库类型
5      'DB_HOST'          => 'localhost',  // 服务器地址
6      'DB_NAME'          => 'test',      // 数据库名
7      'DB_USER'          => 'root',      // 用户名
8      'DB_PWD'           => 'root',      // 密码
9      'DB_PORT'          => '3306',      // 端口
10     'DB_PREFIX'        => '',          // 数据库表前缀
11 );
```

自己安装，安装完以后：访问一下

[http://test\\_thinkphp\\_3.2.3.test/index.php/Home/Index/index](http://test_thinkphp_3.2.3.test/index.php/Home/Index/index)

没报错就是成功

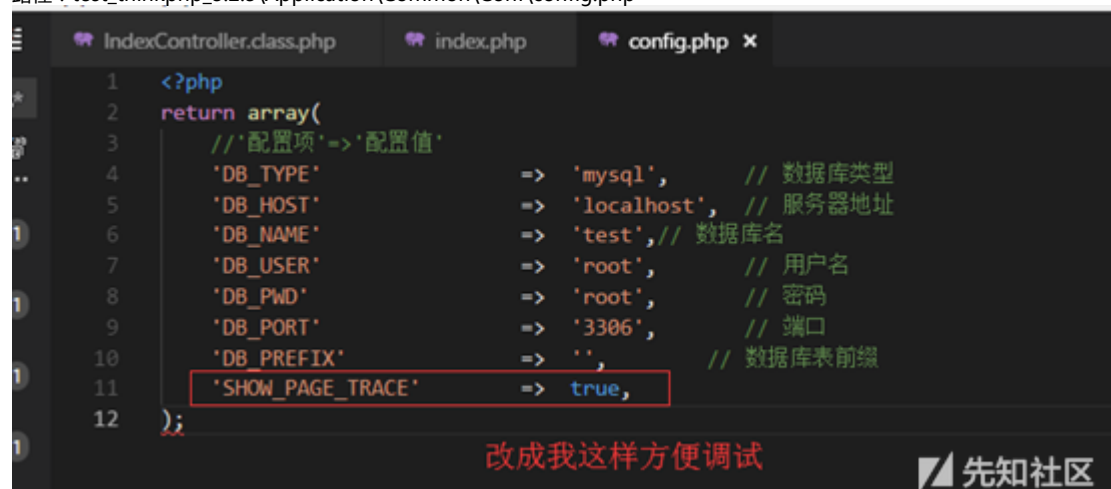
开启debug 方便本地测试

路径：test\_thinkphp\_3.2.3\index.php

```
index.php x
1  <?php
2  // +-----+
3  // | ThinkPHP [ WE CAN DO IT JUST THINK ]
4  // +-----+
5  // | Copyright (c) 2006-2014 http://thinkphp.cn All rights reserved.
6  // +-----+
7  // | Licensed ( http://www.apache.org/licenses/LICENSE-2.0 )
8  // +-----+
9  // | Author: liu21st <liu21st@gmail.com>
10 // +-----+
11
12 // 应用入口文件
13
14 // 检测PHP环境
15 if(version_compare(PHP_VERSION,'5.3.0','<')) die('require PHP > 5.3.0 !');
16
17 // 开启调试模式 建议开发阶段开启 部署阶段注释或者设为false
18 define('APP_DEBUG',True);
19
```

改成我这个样子  
先知社区

路径：test\_thinkphp\_3.2.3\Application\Common\Conf\config.php



```
1 <?php
2 return array(
3     //配置项=>'配置值'
4     'DB_TYPE'           => 'mysql',      // 数据库类型
5     'DB_HOST'           => 'localhost',  // 服务器地址
6     'DB_NAME'           => 'test',      // 数据库名
7     'DB_USER'           => 'root',      // 用户名
8     'DB_PWD'            => 'root',      // 密码
9     'DB_PORT'           => '3306',      // 端口
10    'DB_PREFIX'          => '',          // 数据库表前缀
11    'SHOW_PAGE_TRACE'    => true,
12);
```

改成我这样方便调试

先知社区

0x02 thinkphp3.2 find/select/delete注入演示

3处注入利用方法都是一样的，所以就演示一个 find 注入

Select 与 delete 注入同理



```
// 新0day thinkphp3.2 find/select/delete注入
public function testSqlFind()
{
    $data = M('test')->find(I('GET.test'));
    var_dump($data);

    // M('test')->select(I('GET.test'));

    // M('test')->delete(I('GET.test'));
}
```

测试代码

先知社区



array(4) ( ["id"]=> string(1) "3" ["test"]=> string(3) "123" ["map"]=> NULL ["content"]=> NULL )

基本 文件 流程 错误 SQL 调试

SHOW COLUMNS FROM 'test' [ RunTime:0.0031s ]

SELECT \* FROM 'test' WHERE 'id' = 3 LIMIT 1 [ RunTime:0.0002s ]

test\_thinkphp\_3.2.3.test/index.php/Home/Index/testSqlFind?test=3%27aaa - 360极速浏览器 9.5

域名重定向 test\_thinkphp\_3.2.3.test/index.php/Home/Index/testSqlFind?test=3%27aaa

test\_thinkphp\_3.2.3.test/index.php/Home/Index/testSqlFind?test=3%27aaa

array(4) { ["id"]=> string(1) "3" ["test"]=> string(3) "123" ["map"]=> NULL ["content"]=> NULL }

基本 文件 流程 错误 SQL 调试

SHOW COLUMNS FROM `test` [ RunTime:0.0030s ]

SELECT \* FROM `test` WHERE `id` = 3 LIMIT 1 [ RunTime:0.0002s ] 明显转整了，无法进行注入



test\_thinkphp\_3.2.3.test/index.php/Home/Index/testSqlFind?test[where]=3 - 360极速浏览器 9.5

域名重定向 test\_thinkphp\_3.2.3.test/index.php/Home/Index/testSqlFind?test[where]=3

test\_thinkphp\_3.2.3.test/index.php/Home/Index/testSqlFind?test[where]=3

array(4) { ["id"]=> string(2) "13" ["test"]=> NULL ["map"]=> NULL ["content"]=> NULL }

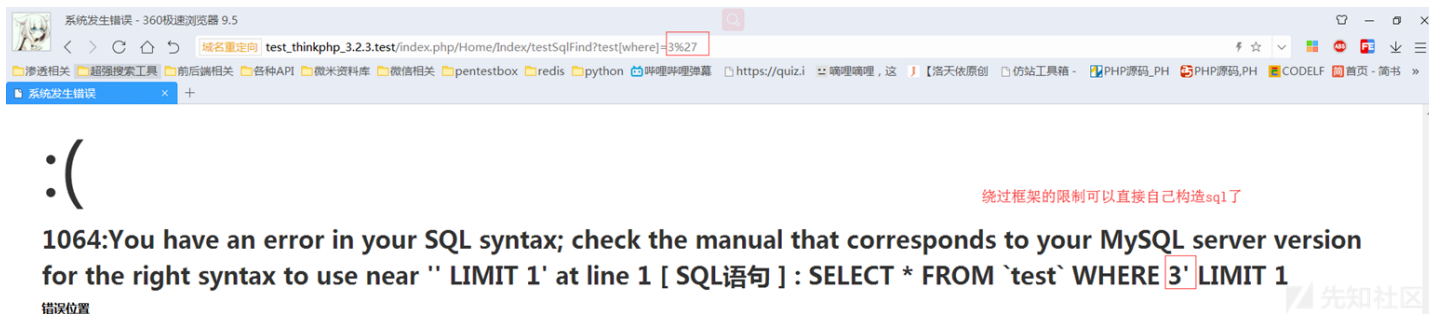
基本 文件 流程 错误 SQL 调试

SHOW COLUMNS FROM `test` [ RunTime:0.0033s ]

SELECT \* FROM `test` WHERE 3 LIMIT 1 [ RunTime:0.0002s ]

可以直接控制where了





#### 0x03 注意点

因为我们例子使用的是TP的I函数而I函数在不指定第三个参数的情况下会默认经过 htmlspecialchars 所以 " > < 都会给转义成实体编码 这就导致了盲注的麻烦

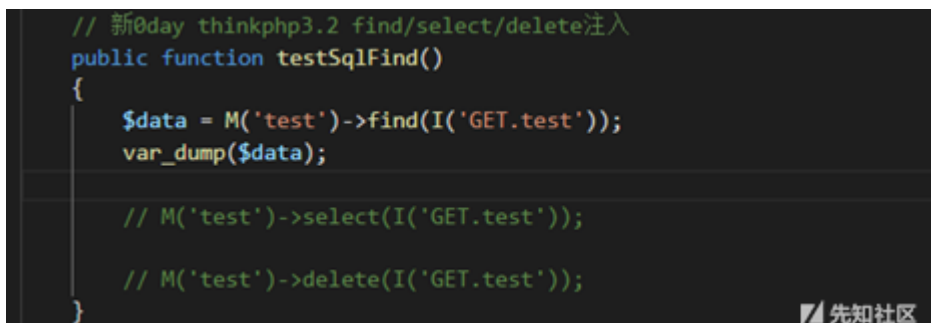
所以如果你们是要盲注的话，就不能使用 " > < "

如果他是指定了第三个参数

例如：

I('GET.test', '', 'trim') 这样指定第三个参数，那么你就可以想怎么注入怎么注入

#### 0x04 漏洞成因



我们先跟进去 find 查看

路径：ThinkPHP\Library\Think\Model.class.php

搜索：function find(

```
714  /**
715   * 查询数据
716   * @access public
717   * @param mixed $options 表达式参数
718   * @return mixed
719   */
720  public function find($options=array()) {
721
722      if(is_numeric($options) || is_string($options)) {
723          $where[$this->getPk()] = $options;
724          $options              = array();
725          $options['where']     = $where;
726      }
727
728      // 根据复合主键查找记录
729      $pk = $this->getPk();
730      if (is_array($options) && (count($options) > 0) && is_array($pk)) {
731          // 根据复合主键查询
732          $count = 0;
733          foreach (array_keys($options) as $key) {
734              if (is_int($key)) $count++;
735          }
736          if ($count == count($pk)) {
737              $i = 0;
738              foreach ($pk as $field) {
739                  $where[$field] = $options[$i];
740                  unset($options[$i++]);
741              }
742              $options['where'] = $where;
743          } else {
744              return false;
745          }
746      }
747      // 总是查找一条记录
748      $options['limit'] = 1;
749      // 分析表达式
750      $options          = $this->_parseOptions($options);
751      // 判断查询缓存
752      if(isset($options['cache'])){
753          $cache = $options['cache'];
```

跟进这里查看即可，上面的都不是重点，也就是说都不会进入那个流程，所以查看 此方法即可



路径：ThinkPHP\Library\Think\Model.class.php

搜索：function \_parseOptions(

```
627  protected function _parseOptions($options=array()) {
628
629      // 这里进行数组合并操作 这里就是注入点
630      if(is_array($options))
631          $options = array_merge($this->options,$options);
632
633      // 获取数据表名
634      if(!isset($options['table'])){
635          // 自动获取表名
636          $options['table'] = $this->getTableName();
637          $fields           = $this->getDbFields();
638      }else{
639          // 指定数据表 则重新获取字段列表 但不支持类型检测
640          $fields           = $this->getDbFields();
641      }
642
643      // 数据表别名
644      if(!empty($options['alias'])){
645          $options['table'] .= ' '.$options['alias'];
646      }
647      // 记录操作的模型名称
648      $options['model'] = $this->name;
649
650      // 字段类型验证
651      if(isset($options['where']) && is_array($options['where']) && !empty($fields) && !isset($options['join'])){
652          // 对数组查询条件进行字段类型检查
653          foreach ($options['where'] as $key=>$val){
654              $key = trim($key);
655              if(in_array($key,$fields,true)){
656                  if(is_scalar($val)) {
657                      $this->parseType($options['where'],$key);
658                  }
659              }elseif(!is_numeric($key) && '.' != substr($key,0,1) && false === strpos($key,'.') && false === strpos($key,'(') && false === strpos($key,')') && false === strpos($key,'|') && false === strpos($key,'&')){
660                  if(empty($this->options['strict'])){
661                      E(L('_ERROR_QUERY_EXPRESS_'),['.$key.'=>'.$val.'']);
662                  }
663                  unset($options['where'][$key]);
664              }
665          }
666      }
667
668      // 查询过后清空sql表达式组装 避免影响下次查询
669      $this->options = array();
670      // 表达式过滤
671      $this->options_filter($options);
672      return $options;
```

先看这里，这里也是我们的漏洞主要触发点，这里进行了一个数组合并，而 \$options 是我们外部可以控制的，我们可以测试一下看看结果



```
1  /**
2   * 分析表达式
3   * @access protected
4   * @param array $options 表达式参数
5   * @return array
6   */
7  protected function _parseOptions($options=array()) {
8
9      // 这里进行数组合并操作 这里就是注入点
10     if(is_array($options))
11         $options = array_merge($this->options,$options);
12     var_dump($options);exit;
13
14     // 获取数据表名
```

调试一波

```
protected function _parseOptions($options=array()) {
    // 这里进行数组合并操作 这里就是注入点
    if(is_array($options))
        $options = array_merge($this->options,$options);

    // 获取数据表名 当$options['table']不存在时执行
    if(!isset($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 当$options['where']为 字符串 并且为 数组 并且 $fields 不为空 并且 $options['join']也存在时进入此流程 (无视即可因为不会进入此流程)
    if(isset($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,true)){
                if(is_scalar($val)) {
                    $this->parseType($options['where'],$key);
                }elseif(!is_numeric($key) && '.' != substr($key,0,1) && false === strpos($key,'.') && false === strpos($key,'(') && false === strpos($key,',') && false === strpos($key,'&')){
                    if(!empty($this->options['strict'])){
                        E(L('_ERROR_QUERY_EXPRESS_').':'. $key.'=>'. $val.'');
                    }
                    unset($options['where'][$key]);
                }
            }
        }
    }

    // 查询过后清空sql表达式组装 避免影响下次查询
    $this->options = array();
    // 表达式过滤
    $this->options_filter($options);
    return $options;
}
```

这里不过多解读，基本的我都已经写了注释了，这里我们简单的认为，他是获取了外部的\$options 然后直接返回了。我们继续查看其他的地方

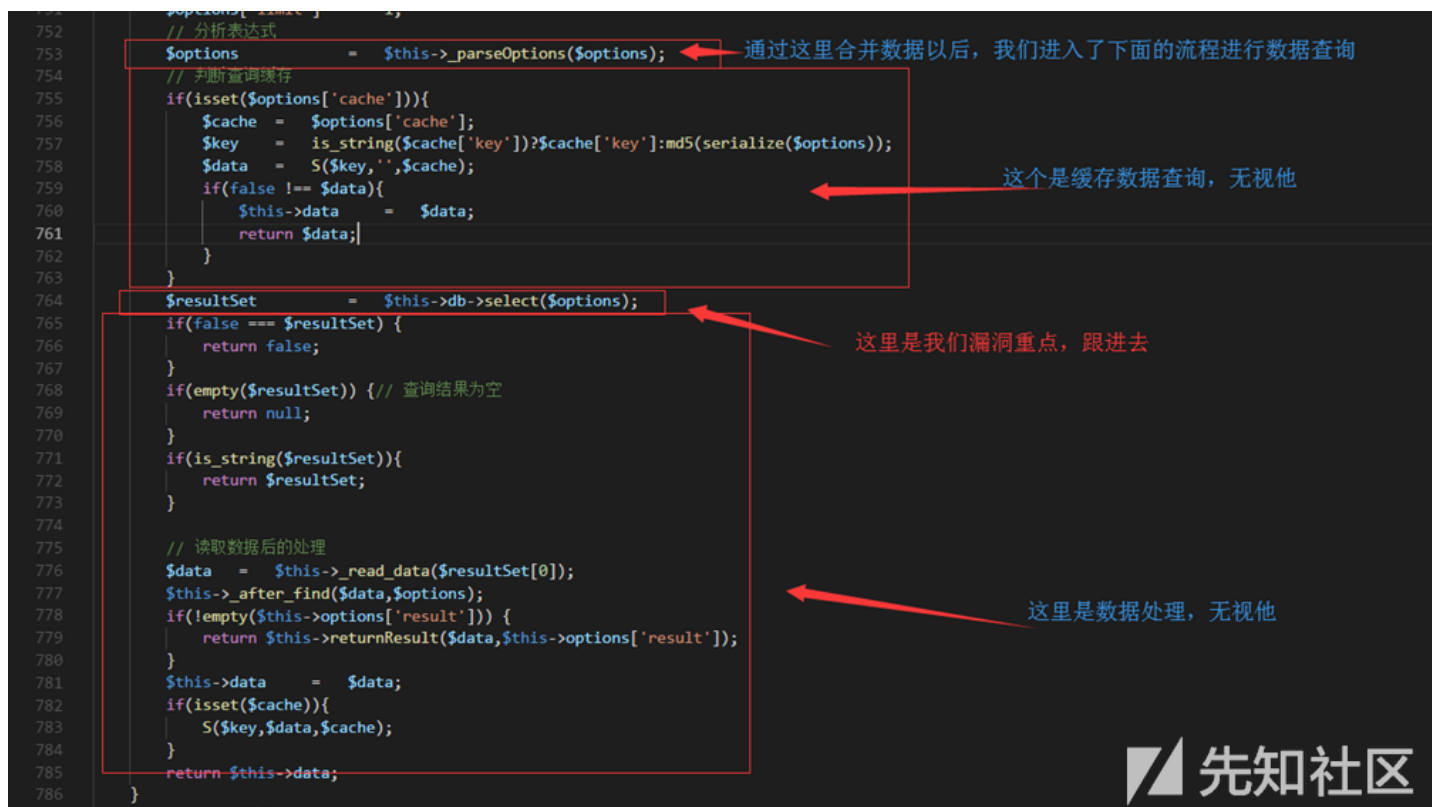
没有方法什么都没执行所以不用去管它

上图中的各种判断，在实际是根本不走的，就算走了也根本无所谓，不影响漏洞使用。所以我们继续跟进

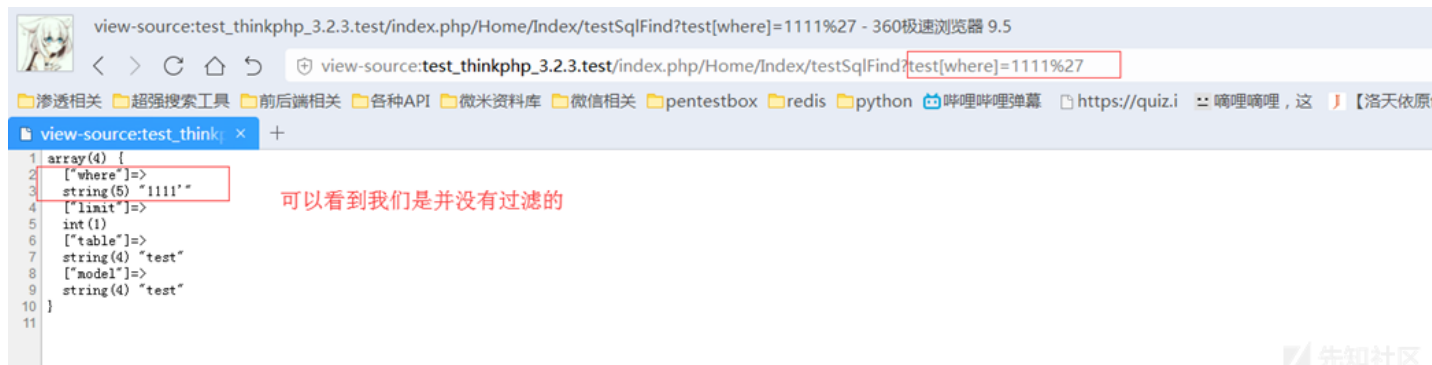
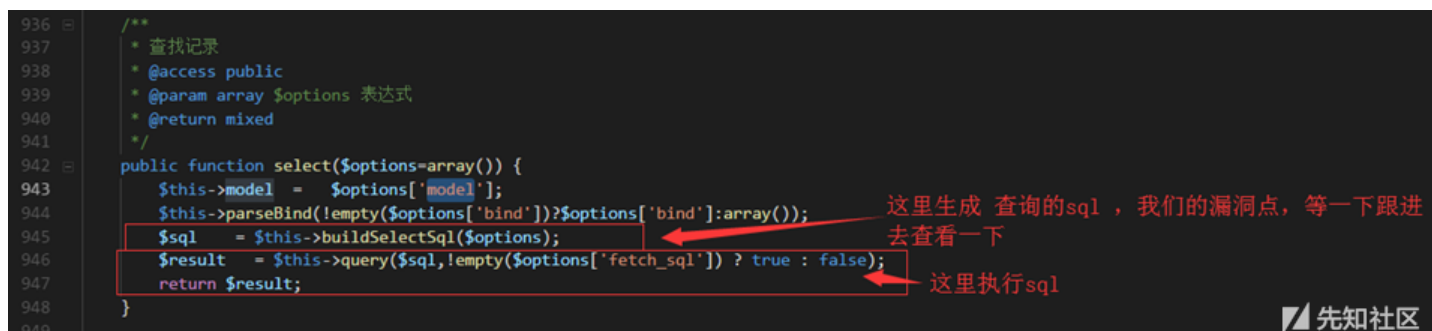
这里重新打开文件：ThinkPHP\Library\Think\Model.class.php

搜索：function find(





打开文件：ThinkPHP\Library\Think\Db\Driver.class.php  
搜索：function select()



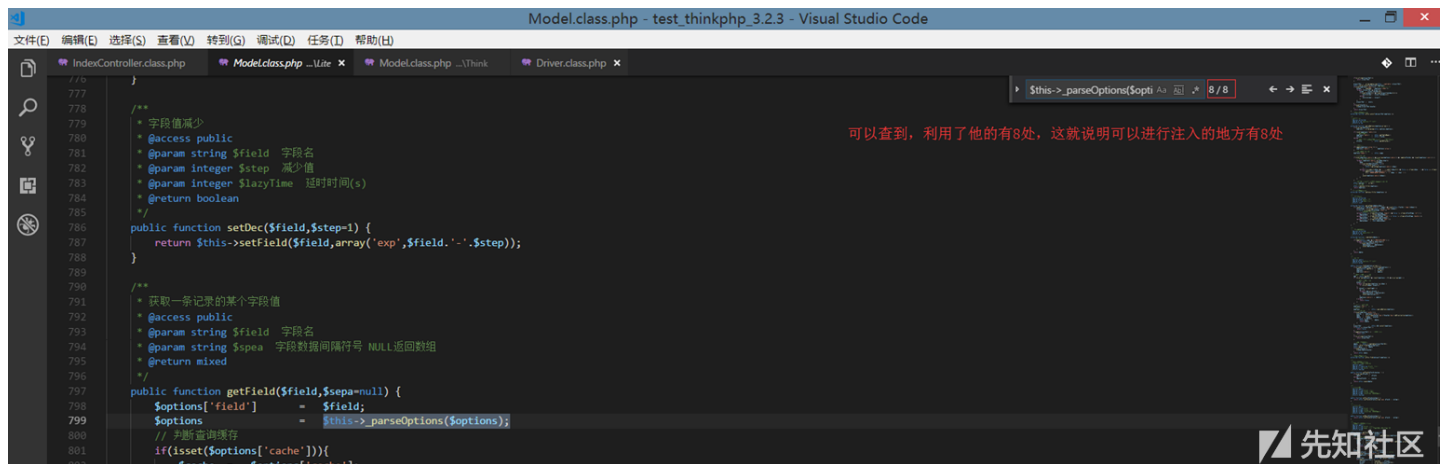
打开文件：ThinkPHP\Library\Think\Db\Driver.class.php

搜索：function parseSql()

```
/**
 * 替换SQL语句中表达式
 * @access public
 * @param array $options 表达式
 * @return string
 */
public function parseSql($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']),
            $this->parseDistinct(isset($options['distinct'])?$options['distinct']:false),
            $this->parseField(empty($options['field'])?$options['field']:''),
            $this->parseJoin(empty($options['join'])?$options['join']:'),
            $this->parseWhere(empty($options['where'])?$options['where']:'),
            $this->parseGroup(empty($options['group'])?$options['group']:'),
            $this->parseHaving(empty($options['having'])?$options['having']:'),
            $this->parseOrder(empty($options['order'])?$options['order']:'),
            $this->parseLimit(empty($options['limit'])?$options['limit']:'),
            $this->parseUnion(empty($options['union'])?$options['union']:'),
            $this->parseLock(isset($options['lock'])?$options['lock']:false),
            $this->parseComment(empty($options['comment'])?$options['comment']:'),
            $this->parseForce(empty($options['force'])?$options['force']:')
        ),$sql);
    return $sql;
}
```

从这里我们可以看到，当options 进入这里以后那么就是直接进行sql拼接了。  
而从这里我们还可以看到，我们注入的时候并不止可以利用 where 还可以直接修改 table 这里举个例子

```
969 /**
970  * 替换SQL语句中表达式
971  * @access public
972  * @param array $options 表达式
973  * @return string
974  */
975 public function parseSql($sql,$options=array()){
976     $sql = str_replace(
977         array('%TABLE%', '%DISTINCT%', '%FIELD%', '%JOIN%', '%WHERE%', '%GROUP%', '%HAVING%', '%ORDER%', '%LIMIT%', '%UNION%', '%LOCK%', '%COMMENT%', '%FORCE%'),
978         array(
979             $this->parseTable($options['table']),
980             $this->parseDistinct(isset($options['distinct'])?$options['distinct']:false),
981             $this->parseField(empty($options['field'])?$options['field']:'),
982             $this->parseJoin(empty($options['join'])?$options['join']:'),
983             $this->parseWhere(empty($options['where'])?$options['where']:'),
984             $this->parseGroup(empty($options['group'])?$options['group']:'),
985             $this->parseHaving(empty($options['having'])?$options['having']:'),
986             $this->parseOrder(empty($options['order'])?$options['order']:'),
987             $this->parseLimit(empty($options['limit'])?$options['limit']:'),
988             $this->parseUnion(empty($options['union'])?$options['union']:'),
989             $this->parseLock(isset($options['lock'])?$options['lock']:false),
990             $this->parseComment(empty($options['comment'])?$options['comment']:'),
991             $this->parseForce(empty($options['force'])?$options['force']:')
992         ),$sql);
993     var_dump($sql);exit;
994     return $sql;
995 }
996
```

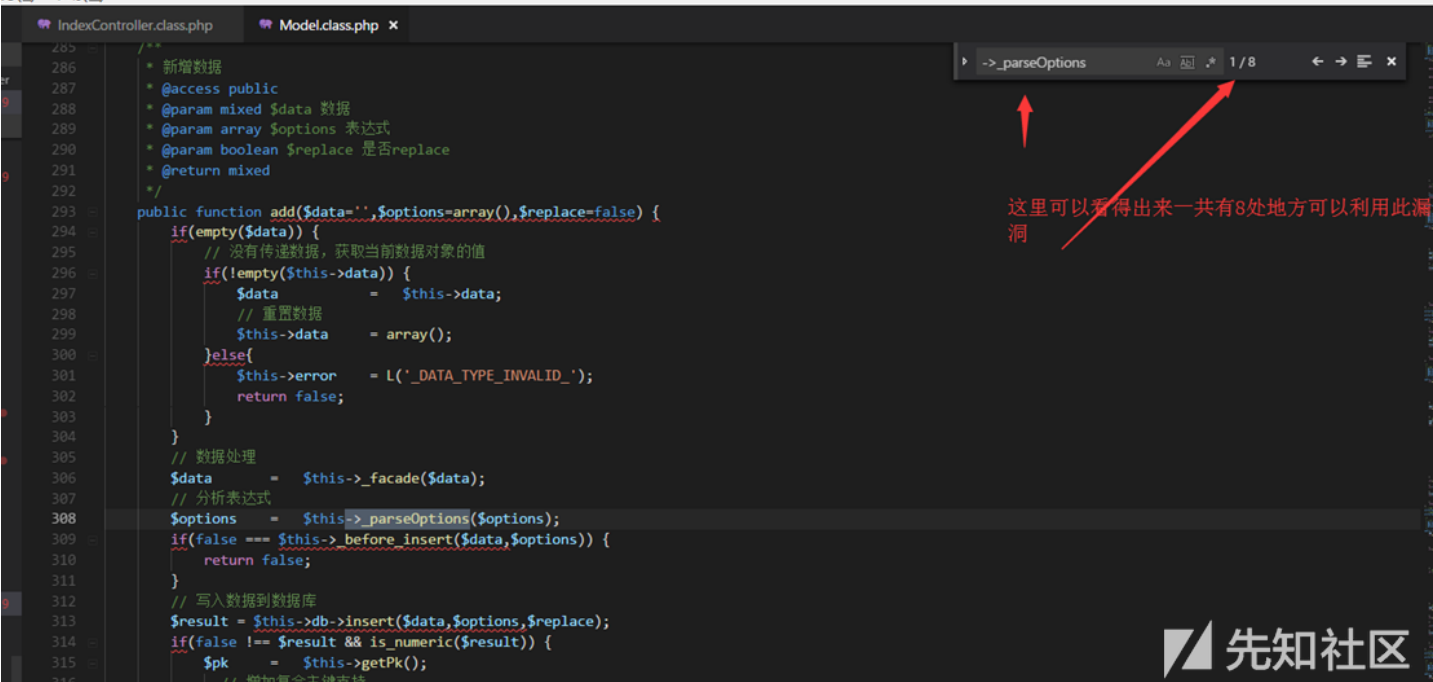


## 0x05 题外话

通过我们前面的一顿分析我们可以得出漏洞的主要原因是因为  
\$this->\_parseOptions(\$options);  
方法进行了 参数合并而最终又没有二次校验导致的任意注入



而论影响的话。  
路径：ThinkPHP\Library\Think\Model.class.php



其中真的可以利用的地方只有6处  
delete 方法 第一个参数可外部控制时可注入  
select 方法 第一个参数可外部控制时可注入  
find 方法 第一个参数可外部控制时可注入  
  
Add 方法 第二个参数可外部控制时可注入  
addAll 方法 第二个参数可外部控制时可注入  
save 方法 第二个参数可外部控制时可注入

水文-thinkphp3.2\_find\_select\_delete注入分析.zip (1.614 MB) [下载附件](#)

点击收藏 | 0 关注 | 1

[上一篇：水文-Thinkphp3.2.3安...](#) [下一篇：某cms前台getshell分析](#)

1. 0 条回复
- 动动手指，沙发就是你的了！

[登录](#) 后跟贴

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)