# 代码审计 | PHPCMS V9 前台 RCE 挖掘分析

本文作者（全网首发）：芝士狍子糕、奶权


## 测试环境

- Nginx 1.4.0
- PHP 5.5.38
- PHPCMS v9 (本漏洞影响全版本)

## 历史漏洞

17 年的时候，php 爆出了一个可以根据随机数种子可被逆推的漏洞。具体的详情以及利用程序在这 MT_RAND SEED CRACKER 对应到旧版本的 PHPCMS 上的利用是：Cookie 前缀 -> 随机数 -> 随机数种子 ->auth_key。具体细节：https://xz.aliyun.com/t/30

## 漏洞修复

### 官方的修复方案是对 Cookie 前缀和 auth_key 生成时使用的种子进行重新播种

```
if($module == 'admin') {
```

```
mt_srand();
$cookie_pre = random(5, 'abcdefghigklmnopqrstuvwxyzABCDEFGHIGKLMNOPQRSTUVWX
mt_srand();
$auth_key = random(20, '1294567890abcdefghigklmnopqrstuvwxyzABCDEFGHIGKLMNO
```



这样的话我们通过 Cookie 前缀生成用的种子就跟 auth_key 生成用的种子不一样了，也就导致 auth_key 的生成序列化不可预测了。这样的修复方法不仔细看貌似没什么问题，但是细想却发现问题很大。PHPCMS 官方对 php 的理解还是不不够深，导致依然可以预测得出 auth_key 生成用的 seed 和 auth_key，这也是本 RCE 漏洞的核心点。

# 源码分析

## PHP 随机数生成分析

## 直接到 Github 上看 php 的源代码：

https://github.com/php/php-src/blob/becda2e0418d4efb55fca40b1170ca67cfbdb4e0/ext/standard/mt_rand.c

```
PHPAPI void php_mt_srand(uint32_t seed)
{
  /* Seed the generator with a simple uint32 */
  php_mt_initialize(seed, BG(state));
  php_mt_reload();
  /* Seed only once */
  BG(mt_rand_is_seeded) = 1;
}
/* }}} */
/* {{{ php_mt_rand
 */
PHPAPI uint32_t php_mt_rand(void)
{
```

```
  /* Pull a 32-bit integer from the generator state
     Every other access function simply transforms the numbers extracted here */
  register uint32_t s1;
  if (UNEXPECTED(!BG(mt_rand_is_seeded))) {
    php_mt_srand(GENERATE_SEED());
  }
  if (BG(left) == 0) {
    php_mt_reload();
  }
  --BG(left);
  s1 = *BG(next)++;
  s1 ^= (s1 >> 11);
  s1 ^= (s1 <<  7) & 0x9d2c5680U;
  s1 ^= (s1 << 15) & 0xefc60000U;
  return ( s1 ^ (s1 >> 18) );
}
```

这里是 php 的 mt_srand 和 mt_rand 的实现，我们这里可以看到，默认情况下的 php_mt_srand 的种子类型是 uint32_t，范围是 2^32-1，数量量并不不大大，我们如果能在本地遍历一遍，并不需要多少时间。也就是说，PHPCMS 这里的修复方式，通过调用多次 mt_srand() 实现重置种子是不安全的。php 默认生成的种子范围是 2^32-1，我们本地机器爆破依然是可以出来的，而他 auth_key 的值并不会改变，只需要电脑上挂机跑即可。

## 漏洞核心原理

假设我们现在拥有了本地暴力遍历 2^32-1 次 seed 的能力，我们改如何利用呢？如果爆破一次就需要往目标服务器上发一次包的话，那利用这个漏洞就需要同时进行 CPU 密集型和 IO 密集型任务，严重拖慢速度不说，还需要往目标发送大量流量，那这个漏洞就没意义了。

所以我们需要找到一个 PHPCMS 内部使用这个 auth_key 进行计算，且计算结果可以被我们获取到的点。而 auth_key 作为 phpcms 的加密 key，最常见的例子就是用来加密 Cookie 了。

我们来看他设置 Cookie 的代码：

我们来看他设置 Cookie 的代码：

```
public static function set_cookie($var, $value = '', $time = 0) {
    $time = $time > 0 ? $time : ($value == '' ? SYS_TIME - 3600 : 0);
    $s = $_SERVER['SERVER_PORT'] == '443' ? 1 : 0;

    $httponly = $var=='userid'||$var=='auth'?true:false;
    $var = pc_base::load_config('system','cookie_pre').$var;
    $_COOKIE[$var] = $value;
    if (is_array($value)) {
      foreach($value as $k=>$v) {
        setcookie($var.'['.$k.']', sys_auth($v, 'ENCODE',
md5(PC_PATH.'cookie'.$var).pc_base::load_config('system','auth_key')), $time,
pc_base::load_config('system','cookie_path'), pc_base::load_config('system','cookie_domain'), $s,
$httponly);
      }
    } else {
      setcookie($var, sys_auth($value, 'ENCODE',
md5(PC_PATH.'cookie'.$var).pc_base::load_config('system','auth_key')), $time,
pc_base::load_config('system','cookie_path'), pc_base::load_config('system','cookie_domain'), $s,
$httponly);
    }
}
```

关注到最后一段，设置的 Cookie 的内容是经过 sys_auth() 进行加密的：

```
function sys_auth($string, $operation = 'ENCODE', $key = '', $expiry = 0) {
  $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_length)) : '';
  $cryptkey = $keya.md5($keya.$keyc);
  $key_length = strlen($cryptkey);
  $string = $operation == 'DECODE' ? base64_decode(strtr(substr($string, $ckey_length), '-_', '+/'))
: sprintf('%010d', $expiry ? $expiry + time() : 0).substr(md5($string.$keyb), 0, 16).$string;
```

```
  $string_length = strlen($string);
  $result = '';
  $box = range(0, 255);
  $rndkey = array();
  for($i = 0; $i <= 255; $i++) {

    $rndkey[$i] = ord($cryptkey[$i % $key_length]);
  }
  for($j = $i = 0; $i < 256; $i++) {
    $j = ($j + $box[$i] + $rndkey[$i]) % 256;
    $tmp = $box[$i];
    $box[$i] = $box[$j];
    $box[$j] = $tmp;
  }
  for($a = $j = $i = 0; $i < $string_length; $i++) {
    $a = ($a + 1) % 256;
    $j = ($j + $box[$a]) % 256;
    $tmp = $box[$a];
    $box[$a] = $box[$j];
    $box[$j] = $tmp;
    $result .= chr(ord($string[$i]) ^ ($box[($box[$a] + $box[$j]) % 256]));
  }
  if($operation == 'DECODE') {
    if((substr($result, 0, 10) == 0 || substr($result, 0, 10) - time() > 0) && substr($result, 10,
16) == substr(md5(substr($result, 26).$keyb), 0, 16)) {
      return substr($result, 26);
    } else {
      return '';
    }
  } else {
    return $keyc.rtrim(strtr(base64_encode($result), '+/', '-_'), '=');
  }
}
```

经常审代码的人应该一眼就看出来了，这个函数是直接复制了 Discuz 中的一个流加密算法。该函数可以利用一个 key 来对数据进行加解密。我们往回看，程序中往这个函数传递了什么 key：

```
sys_auth($value, 'ENCODE', md5(PC_PATH.'cookie'.$var).pc_base::load_config('system','auth_key'))
```

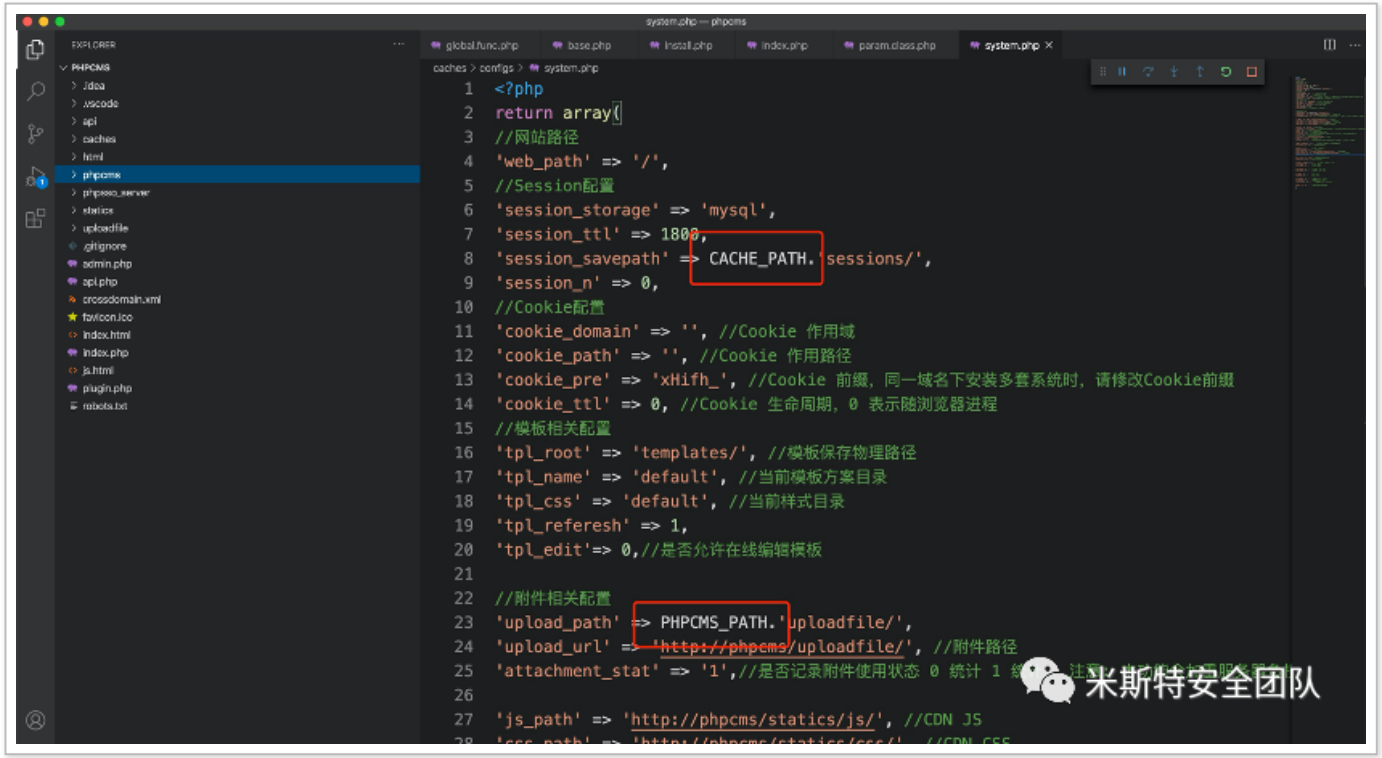
key 的生成依赖以下部分：

1. PC_PATH
2. $var
3. auth_key

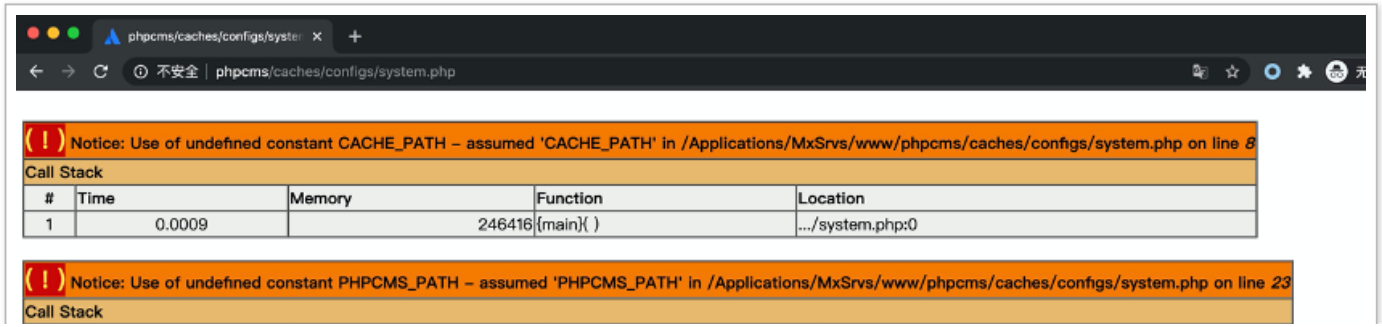
$var 是当前设置 Cookie 的键我们可以轻松从响应头中获取到。auth_key 经过我们前面的分析可以知道，它其实本质上就是一个 2^32-1 这个范围内的数字。最后一个未知量为 PC_PATH，我们搜索一下看一下其定义：

```
//PHPCMS框架路径
define('PC_PATH', dirname(__FILE__).DIRECTORY_SEPARATOR);
```

实际上就是 WEB 的路径，也就是说我们利用这个漏洞还需要搭配一个 WEB 路径泄露。所幸，PHPCMS 的开发习惯并不好，很多地方都没有定义变量和限制访问，比如 /caches/configs/system.php 文件：


system.php — phpcms
EXPLORER
PHPCMS
global.func.php
base.php
install.php
index.php
param.class.php
system.php
caches > configs > system.php
<?php
return array(
//网站路径
'web_path' => '/',
//Session配置
'session_storage' => 'mysql',
'session_ttl' => 1800,
'session_savepath' => CACHE_PATH.'sessions/',
'session_n' => 0,
//Cookie配置
'cookie_domain' => '', //Cookie 作用域
'cookie_path' => '', //Cookie 作用路径
'cookie_pre' => 'xHifh_', //Cookie 前缀，同一域名下安装多套系统时，请修改Cookie前缀
'cookie_ttl' => 0, //Cookie 生命周期，0 表示随浏览器进程
//模板相关配置
'tpl_root' => 'templates/', //模板保存物理路径
'tpl_name' => 'default', //当前模板方案目录
'tpl_css' => 'default', //当前样式目录
'tpl_referesh' => 1,
'tpl_edit'=> 0,//是否允许在线编辑模板
//附件相关配置
'upload_path' => PHPCMS_PATH.'uploadfile/',
'upload_url' => 'http://phpcms/uploadfile/', //附件路径
'attachment_stat' => '1',//是否记录附件使用状态 0 统计 1
'js_path' => 'http://phpcms/statics/js/', //CDN JS
米斯特安全团队

直接访问会报错：


phpcms/caches/configs/syste
不安全 | phpcms/caches/configs/system.php
Notice: Use of undefined constant CACHE_PATH – assumed 'CACHE_PATH' in /Applications/MxSrvs/www/phpcms/caches/configs/system.php on line 8
Call Stack
# Time Memory Function Location
1 0.0009 246416 {main}( ) .../system.php:0
Notice: Use of undefined constant PHPCMS_PATH – assumed 'PHPCMS_PATH' in /Applications/MxSrvs/www/phpcms/caches/configs/system.php on line 23
Call Stack

| # | Time | Memory | Function | Location |
|---|---|---|---|---|
| 1 | 0.0009 | 246416 | {main}( ) | .../system.php:0 |



现在爆破这个 auth_key 的思路就出来了

1. 通过路径泄漏拿到WEB路径
2. 通过响应头拿到设置的Cookie的键
3. 不断从uint32_t*范围生成的序列中取值，生成对应种子下的*auth_key
4. 将从响应头中拿到的设置的Cookie的内容放进流加密函数中进行解密，使用的key为上面1-3根据对应规则生成的值。
5. 判断解密后的字符串是否与Cookie加密前的字符串相等

现在所有的利用链就差最后一个点了，就是找到一个通过 set_cookie() 加密的一个 Cookie 即可。这个条件在 PHPCMS 中比比皆是，我们随便找一个不依赖用户中心，不依赖任何需要验证权限的模块的地方即可，比如：phpcms/modules/mood/index.php 中的 post() 函数

```
public function post() {
    if (isset($_GET['callback']) && !preg_match('/^[a-zA-Z_][a-zA-Z0-9_]+$/', $_GET['callback']))
unset($_GET['callback']);
    $mood_id =& $this->mood_id;
    $setting =& $this->setting;
    $cookies = param::get_cookie('mood_id');
    $cookie = explode(',', $cookies);
    if (in_array($this->mood_id, $cookie)) {
      $this->_show_result(0, L('expressed'));
    } 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->contentid))) {
```

```
        $mood_db->update(array('total'=>'+=1', $fields=>'+=1', 'lastupdate'=>SYS_TIME),
array('id'=>$data['id']));
        $data['total']++;
        $data[$fields]++;
      } else {

        $mood_db->insert(array('total'=>'1', $fields=>'1', 'catid'=>$this->catid, 'siteid'=>$this-
>siteid, 'contentid'=>$this->contentid,'
        lastupdate'=>SYS_TIME));
        $data['total'] = 1;
        $data[$fields] = 1;
      }
      param::set_cookie('mood_id', $cookies.','.$mood_id);
      foreach ($setting as $k=>$v) {
        $setting[$k]['fields'] = 'n'.$k;
        if (!isset($data[$setting[$k]['fields']])) $data[$setting[$k]['fields']] = 0;
        if (isset($data['total']) && !empty($data['total'])) {
          $setting[$k]['per'] = ceil(($data[$setting[$k]['fields']]/$data['total']) * 60);
        } else {
          $setting[$k]['per'] = 0;
        }
      }
      ob_start();
      include template('mood', 'index');
      $html = ob_get_contents();
      ob_clean();
      $this->_show_result(1,$html);
    }
}
```

其中 set_cookie('mood_id',$cookies.','.$mood_id); 里的 $mood_id 在构造函数里赋值

```
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);
}
```

可以看到可以通过我们 get 传过去的 id 值进行控制，获取 Cookie 的请求如下：

```
req:
GET /index.php?m=mood&c=index&a=post&k=1&id=nqtest HTTP/1.1
Host: phpcms
Connection: close
res:
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Connection: close
Set-Cookie: xHifh_mood_id=09a5aQTLUUxyVdgD66CHwOsEbFrKP0Ztaxs1T1ltBrVn4Z2X
Content-Length: 1000
(略)
```

通过这个请求我们就可以拿到 Cookie 的键与 nqtest 加密过后的值了，已经满足了利用条件。

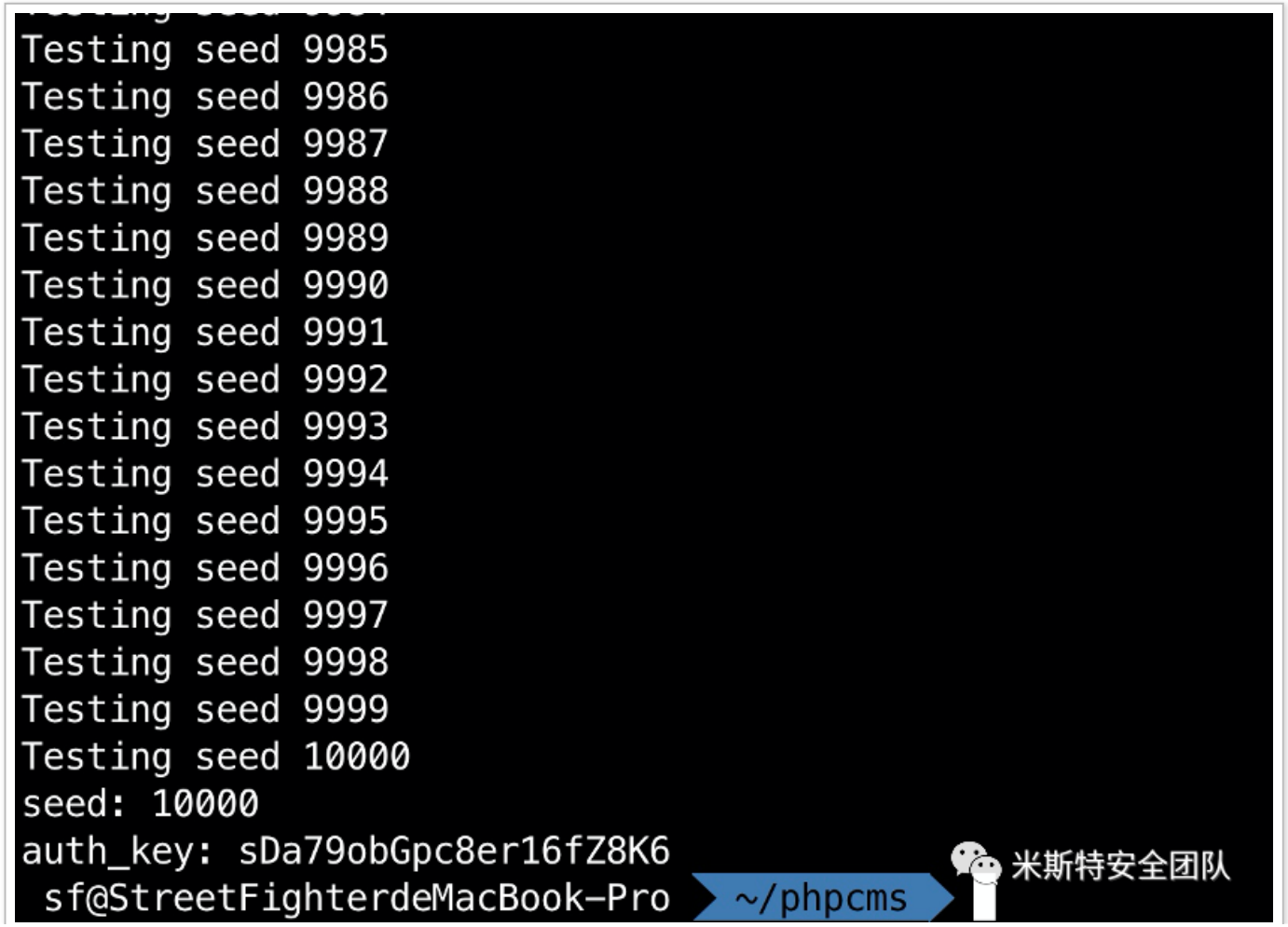
# POC

简单写了一个 POC 来进行测试

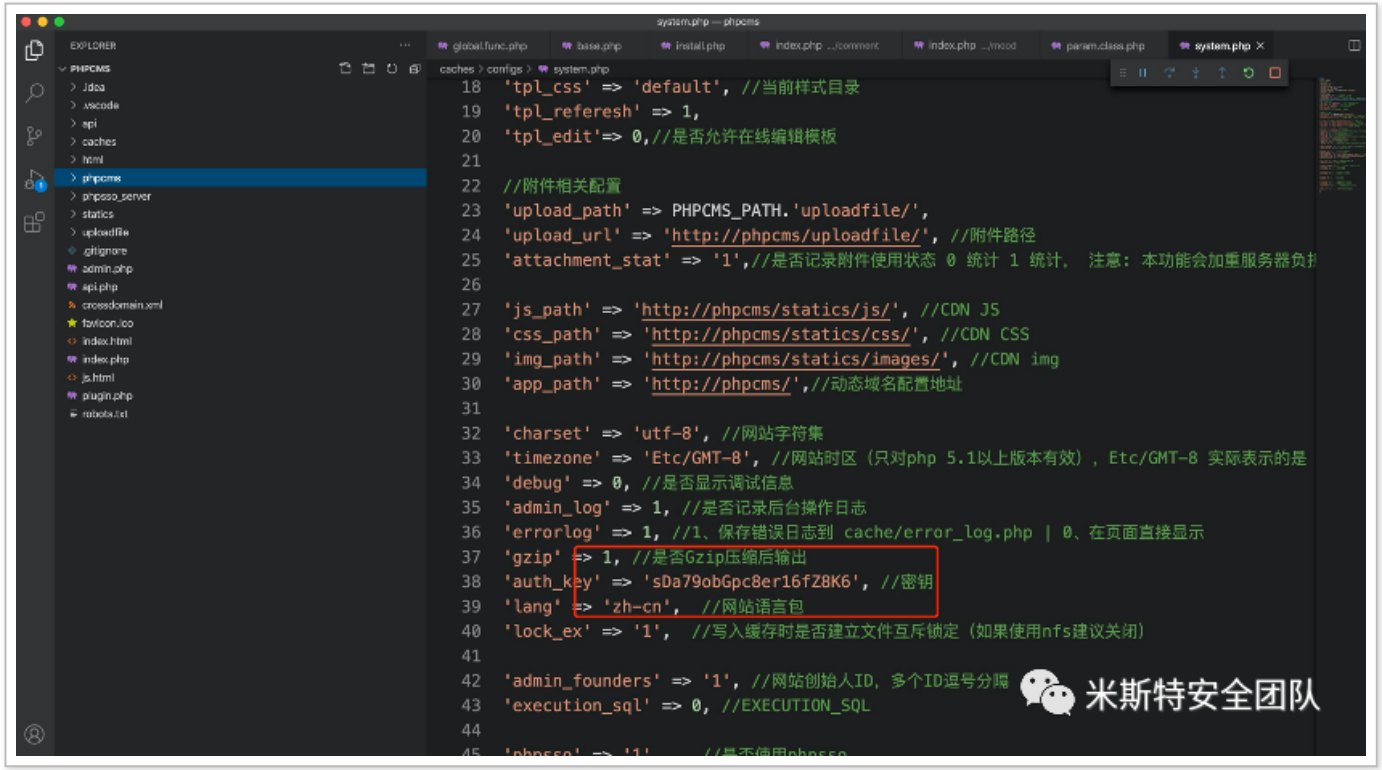
```
<?php
function sys_auth($string, $operation = 'ENCODE', $key = '', $expiry = 0) {
  $ckey_length = 4;
  $key = md5($key != '' ? $key : pc_base::load_config('system', 'auth_key'));
```

```
  $keya = md5(substr($key, 0, 16));
  $keyb = md5(substr($key, 16, 16));
  $keyc = $ckey_length ? ($operation == 'DECODE' ? substr($string, 0, $ckey_length):
substr(md5(microtime()), -$ckey_length)) : '';
  $cryptkey = $keya.md5($keya.$keyc);
  $key_length = strlen($cryptkey);
  $string = $operation == 'DECODE' ? base64_decode(strtr(substr($string, $ckey_length), '-_', '+/'))
: sprintf('%010d', $expiry ? $expiry + time() : 0).substr(md5($string.$keyb), 0, 16).$string;
  $string_length = strlen($string);
  $result = '';
  $box = range(0, 255);
  $rndkey = array();
  for($i = 0; $i <= 255; $i++) {
    $rndkey[$i] = ord($cryptkey[$i % $key_length]);
  }
  for($j = $i = 0; $i < 256; $i++) {
    $j = ($j + $box[$i] + $rndkey[$i]) % 256;
    $tmp = $box[$i];
    $box[$i] = $box[$j];
    $box[$j] = $tmp;
  }
  for($a = $j = $i = 0; $i < $string_length; $i++) {
    $a = ($a + 1) % 256;
    $j = ($j + $box[$a]) % 256;
    $tmp = $box[$a];
    $box[$a] = $box[$j];
    $box[$j] = $tmp;
    $result .= chr(ord($string[$i]) ^ ($box[($box[$a] + $box[$j]) % 256]));
  }
  if($operation == 'DECODE') {
    if((substr($result, 0, 10) == 0 || substr($result, 0, 10) - time() > 0) && substr($result, 10,
16) == substr(md5(substr($result, 26).$keyb), 0, 16)) {
      return substr($result, 26);
    } else {
      return '';
    }
  } else {
    return $keyc.rtrim(strtr(base64_encode($result), '+/', '-_'), '=');
```

```
  }
}
function random($length, $chars = '0123456789', $seed) {
  $hash = '';
  $max = strlen($chars) - 1;
  mt_srand($seed);
  for($i = 0; $i < $length; $i++) {
    $hash .= $chars[mt_rand(0, $max)];
  }
  return $hash;
}
// 完整POC请看下方
```



可以看到跑出来的 auth_key 是正确的




## 可行性探究

上面的复现过程其实偷了个懒，手动在生成 auth_key 之前对接下来的随机数序列播了种

```
function random($length, $chars = '0123456789') {
  $hash = '';
  $max = strlen($chars) - 1;
  mt_srand(10000);
  for($i = 0; $i < $length; $i++) {
```

```
    for($i = 0; $i < $length; $i++) {
      $hash .= $chars[mt_rand(0, $max)];
    }
    return $hash;
}
```

根据大概的统计，上面使用 php 实现的 POC 跑完大概需要将近 15 天左右的时间，所以我这里为了证明可行性手动播了种。

实战利用的话可以采用分布式的思想，一台机器跑一部分序列，理论上只要机器足够多，跑出 auth_key 的速度就能足够快。为了让速度更快，我将上面的 POC 实现了一个基于 C 的 EXP 版本，可以将速度提升到 3 天内跑完，如果加上多线程与分布式，则利用到目标上是完全可以实现的。

完整的 POC 与 EXP 请关注我的知识星球

## 后续利用

至于获取到 auth_key 后可以做一些什么这个大家都明白，网上也有一大把的分析，这里就不再赘述了。