[红日安全]代码审计Day9 - str_replace函数过滤不当

红日安全 / 2018-08-24 21:10:22 / 浏览数 3688 安全技术 漏洞分析 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

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9篇 代码审计文章：

Day 9 - Rabbit

题目叫做兔子，代码如下：

```
1 class LanguageManager
2 {
3     public function loadLanguage()
4     {
5         $lang = $this->getBrowserLanguage();
6         $sanitizedLang = $this->sanitizeLanguage($lang);
7         require_once("/lang/$sanitizedLang");
8     }
9
10    private function getBrowserLanguage()
11    {
12        $lang = $_SERVER['HTTP_ACCEPT_LANGUAGE'] ?? 'en';
13        return $lang;
14    }
15
16    private function sanitizeLanguage($language)
17    {
18        return str_replace('../', '', $language);
19    }
20 }
21
22 (new LanguageManager())->loadLanguage();
```


漏洞解析 ：

这一题考察的是一个 str_replace 函数过滤不当造成的任意文件包含漏洞。在上图代码 第18行 处，程序仅仅只是将 ../ 字符替换成空，这并不能阻止攻击者进行攻击。例如攻击者使用payload：..../// 或者 ..././ ，在经过程序的 str_replace 函数处理后，都会变成 ../ ，所以上图程序中的 str_replace 函数过滤是有问题的。我们来看一下PHP手册对 str_replace 函数的具体定义：

> str_replace ：(PHP 4, PHP 5, PHP 7)
>
> 功能 ：子字符串替换
>
> 定义 ： `mixed str_replace ( mixed $search , mixed $replace , mixed $subject [, int &$count ] )`
>
> 该函数返回一个字符串或者数组。如下：
>
> str_replace(字符串1，字符串2，字符串3)：将字符串3中出现的所有字符串1换成字符串2。
>
> str_replace(数组1，字符串1，字符串2)：将字符串2中出现的所有数组1中的值，换成字符串1。
>
> str_replace(数组1，数组2，字符串1)：将字符串1中出现的所有数组1一一对应，替换成数组2的值，多余的替换成空字符串。

```php
<?php
    $string = "blue white grey yellow green puple red orange black";
    echo $string."<br>";
    echo str_replace('grey', 'pink' , $string);
    echo "<br>";
    echo str_replace(array('blue','white','grey'), 'pink' , $string);
    echo "<br>";
    echo str_replace(array('blue','white'), array('hello','world') , $string);
    /* 输出结果
     * blue white grey yellow green puple red orange black
     * blue white pink yellow green puple red orange black
     * pink pink pink yellow green puple red orange black
     * hello world grey yellow green puple red orange black
     */
?>
```


实例分析

本次实例分析，我们选取的是 Metinfo 6.0.0 版本。漏洞文件在 app/system/include/module/old_thumb.class.php 中，我们发现程序将变量 $dir 中出现的 ../ 和 ./ 字符替换成空字符串（下图第6行处），猜想开发者应该是有考虑到路径穿越问题，所以做了此限制。具体代码如下：

```php
<?php
public function doshow()
{
    global $_M;

    $dir = str_replace(array('../', './'), '', $_GET['dir']);

    if (strstr(str_replace($_M['url']['site'], '', $dir), 'http')) {
        header("Content-type: image/jpeg");
        ob_start();
        readfile($dir);
        ob_flush();
        flush();
        die;
    }
    ......
}
```


接着在第8行处，用 strstr 函数判断 $dir 变量中是否含有 http 字符串，如果有，则读取加载 $dir
变量，并以图片方式显示出来。这里猜测开发者的意图是，加载远程图片。关于 strstr 函数，定义如下：

strstr ：(PHP 4, PHP 5, PHP 7)

功能 ：查找字符串的首次出现

定义 ： `string strstr ( string $haystack , mixed $needle [, bool $before_needle = FALSE ] )`

返回 haystack 字符串从 needle 第一次出现的位置开始到 haystack 结尾的字符串。

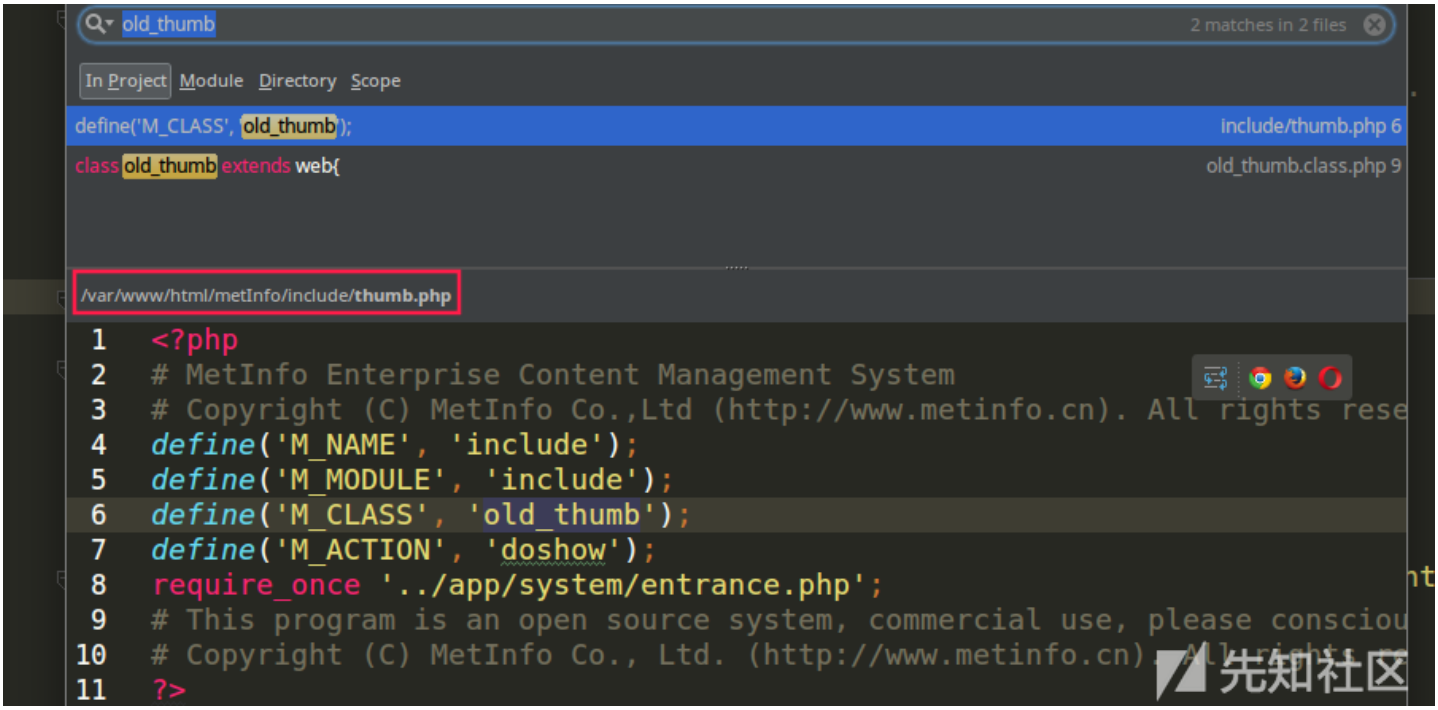
```
domain = strstr('hongrisec@gmail.com', '@');
// ■■■■■@gmail.com
user = strstr('hongrisec@gmail.com, '@', true); // ■ PHP 5.3.0 ■
// ■■■■■hongrisec
```

然而这段代码是可以绕过的，例如我们使用 payload：...../..//http/...../..//...../..//...../..//...../..//etc/passwd ，过滤后实际就变成： ../http/../../../../etc/passwd ，效果如下：

```php
<?php
$dir = str_replace(array('../','./'), '', $_GET['dir']);

if(strstr($dir, 'http')){
    echo readfile($dir);
}
else{
    die("Invalid dir!");
}
```

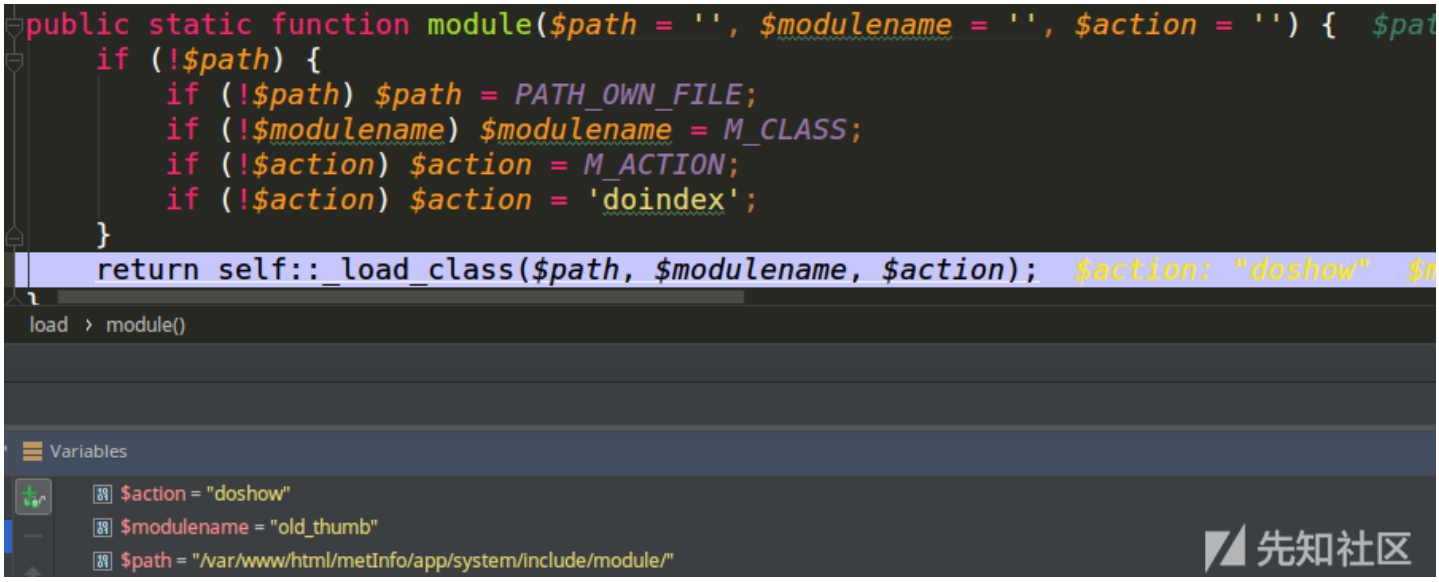
接下来，我们要做的就是搜索程序在哪里调用了这个文件。用 phpstorm 加载整个项目文件，按住 `Ctrl+Shift+F` 键，搜索关键词 old_thumb ，发现在 include/thumb.php 文件中调用 old_thumb 类，搜索结果如下图：



我们在 include/thumb.php 文件中，可以看到 M_CLASS 定义为 old_thumb ，而 M_ACTION 定义为 doshow 。我们接着跟进到 app/system/entrance.php 文件中，在该文件的末尾可以看包含了 app/system/include/class/load.class.php 文件，引入了 load 类，然后调用了 load 类的 module 方法。

```
// app/system/include/class/load.class.php
require_once PATH_SYS_CLASS.'load.class.php';
load::module();
```

我们跟进 module 方法，并查看各个变量的赋值情况(app/system/include/class/load.class.php 文件)：



上图程序最后调用了 load 类的 _load_class 方法，我们跟进该方法，详细代码如下：

```
1 private static function _load_class($path, $classname, $action = '') {
2     $classname=str_replace('.class.php', '', $classname);
3     $is_myclass = 0;
4     if(!self::$mclass[$classname]){
5         if(file_exists($path.$classname.'.class.php')){
6             require_once $path.$classname.'.class.php';
7         }// require_once 'app/system/include/module/old_thumb.class.php'
8         ......
9     }
10    if ($action) { // $action=doshow
11        ......
12        else{
13            if($is_myclass){
14                $newclass = new $myclass;
15            }else{
16                $newclass = new $classname;//new old_thumb
17            }
18            self::$mclass[$classname] = $newclass;
19        }
20        if ($action!='new') {
21            if(substr($action, 0, 2) != 'do'){
22                die($action.' function no permission load!!!');
23            }
24            if(method_exists($newclass, $action)){
25                call_user_func(array($newclass, $action));//调用old_thumb类的doshow方法
26            ......
```


可以看到上图代码第16行处实例化了一个 old_thumb 类对象，然后在第25行处调用了 old_thumb 类的 doshow 方法， doshow 方法中的 $dir 变量就是用户可以控制的。以上便是完整的攻击过程分析，下面我们看看具体如何进行攻击。

漏洞利用

实际上攻击的话就很简单了，因为 $dir 变量是直接通过 GET请求 获取的，然后用 str_replace 方法处理，而 str_replace 方法处理又有问题，所以我们构造 payload 如下：

```
http://localhost/metInfo/include/thumb.php?dir=...../////http/...../////■■■■■■■■■■.txt
```

成功读取 最终用户授权许可协议.txt 文件。

修复建议

关于修复建议，这里先抛出个问题给大家，针对这个案例，下面的修复代码是否可行？

```
$dir = str_replace(array('..','//'), '', $_GET['dir']);
```

咋一看，这个代码好像完美地修复了路径穿越问题，但是，我们在修复代码的时候一定要结合实际情况。比如在metinfo中，程序这里原来的功能是加载远程图片，使用上面

```php
$dir = str_replace('..', '', $dir = $_GET['path']);
if(stripos($dir, 'http://')===0 or stripos($dir, 'https://')===0){
    header("Content-type: image/jpeg");
    ob_start();
    readfile($dir);
    ob_flush();
    flush();
    die;
}
else die("Hacker found!");
```


结语

看完了上述分析，不知道大家是否对 str_replace() 函数过滤路径符号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。Day9
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
<?php
include 'config.php';
include 'function.php';

$conn = new mysqli($servername,$username,$password,$dbname);
if($conn->connect_error){
   die('■■■■■■■■');
}

$sql = "SELECT COUNT(*) FROM users";
$result = $conn->query($sql);
if($result->num_rows > 0){
   $row = $result->fetch_assoc();
   $id = $row['COUNT(*)'] + 1;
}
else die($conn->error);

if(isset($_POST['msg']) && $_POST['msg'] !==''){
   $msg = addslashes($_POST['msg']);
   $msg = replace_bad_word(convert($msg));
   $sql = "INSERT INTO users VALUES($id,'".$msg."')";
```

```
   $result = $conn->query($sql);
   if($conn->error) die($conn->error);
}
echo "<center><h1>Welcome come to HRSEC message board</center></h1>";
echo <<<EOF
<center>
   <form action="index.php" method="post">
       <p>Leave a message: <input type="text" name="msg" /><input type="submit" value="Submit" /></p>
   </form>
</center>
EOF;
$sql = "SELECT * FROM users";
$result = $conn->query($sql);
if($result->num_rows > 0){
   echo "<center><table border='1'><tr><th>id</th><th>message</th><tr></center>";
   while($row = $result->fetch_row()){
       echo "<tr><th>$row[0]</th><th>$row[1]</th><tr>";
   }
   echo "</table></center>";
}
$conn->close();
?>

// function.php
<?php
function replace_bad_word($str){
   global $limit_words;
   foreach ($limit_words as $old => $new) {
       strlen($old) > 2 && $str = str_replace($old,trim($new),$str);
   }
   return $str;
}

function convert($str){
   return htmlentities($str);
}

$limit_words = array('■■' => '■**', '■■■' => '■**');

foreach (array('_GET','_POST') as $method) {
   foreach ($$method as $key => $value) {
       $$key = $value;
   }
}
?>

// config.php
<?php
$servername = "localhost";
$username = "hongrisec";
$password = "hongrisec";
$dbname = "day9";
?>

# ■■CTF■■■■■sql■■
create database day9;
use day9;
create table users(
id integer auto_increment not null primary key,
message varchar(50)
);
create table flag( flag varchar(40));
insert into flag values('HRCTF{StR_R3p1ac3_anD_sQ1_inJ3ctIon_zZz}');
```

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

Metinfo 6.0.0 任意文件读取漏洞

MetInfo 任意文件读取漏洞的修复与绕过

点击收藏 | 0 关注 | 1

上一篇：某cms前台getshell分析 下一篇：基于HTTP Referer头部的...

1. 5 条回复

239947****@qq.co 2018-08-25 23:33:24

坑爹的作者发一个cms都粗心大意，关键代码哪一样都不一样。好几次都这样。

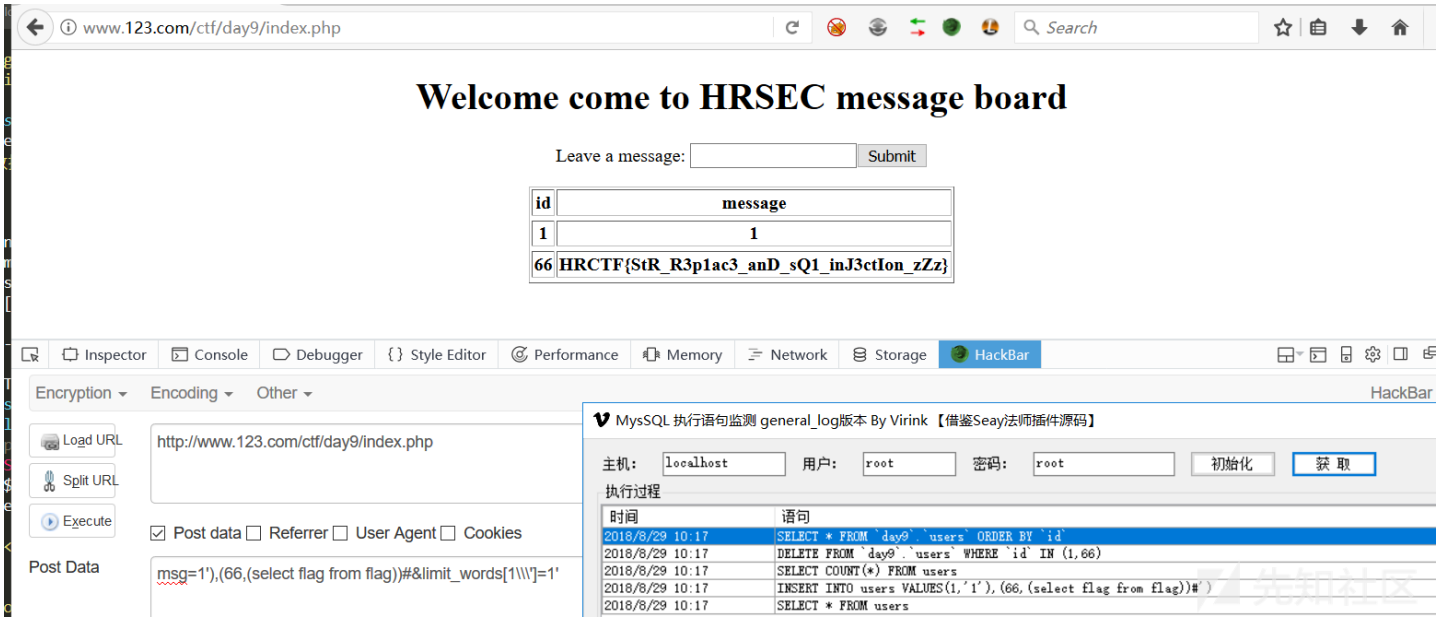
0 回复Ta

红日安全 2018-08-26 10:12:30

@239947****@qq.co 哪里不一样，还望您具体指出。这篇文章很早就写了，我看了一下文中的CMS下载链接，代码没有问题

0 回复Ta

afanti 2018-08-29 10:18:12



0 回复Ta

红日安全 2018-08-30 10:18:32

@afanti 厉害了，这题实际上就是根据齐博CMS的一个漏洞改编的。

0 回复Ta

roothex 2019-08-16 10:26:16

@红日安全 文中strstr函数在CMS里是substr

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