ZDI年度五大漏洞第五弹—ZDI-18-1372，巧妙的绕过

Agostop / 2018-12-29 10:50:00 / 浏览数 2325 技术文章 翻译文章 顶(0) 踩(0)

今年十月，我在博客中记录了一个不久前在Adobe Reader DC中发现的新的攻击面,可以通过Catalog插件实现攻击，这个插件暴露了一组很好用的JavaScript API。对于我提交的这个漏洞，Adobe不仅修补了相关的解析漏洞，还禁用了Catalog插件中能够触发文件格式解析操作的相关JavaScript API。

实际上，解析代码在Adobe Reader中是仍然启用的，只是触发解析代码的过程不像以前那么直接。我很好奇解析器中是否还存在具有足够研究价值的bug，我甚至还快速浏览了一遍，仔细考虑了所有可 API绕过研究，然而，如果当时早知道3年后我可以像几周前那样绕过JavaScript API的限制，那么我肯定会更认真地对待这个索引问题。

无论如何，当Catalog漏洞公开时，Sebastian Apelt (@bitshifter123) 主动联系了我并提到他也在研究这个索引攻击面。有趣的是，Sebastian是通过Search插件而不是Catalog插件来到达攻击面。Sebastian研究的有趣之处在于，他找到了一种

绕过

在Adobe Acrobat JavaScript中，“search”对象的“query”方法被标注了“S”级别的安全限制。通常，有四种方法可以从JavaScript中调用这个方法：

1. search.query(<word>) </word>
2. search.query(<word>, "Index", <idx-path>) </idx-path></word>
3. search.query(<word>, "Folder", <idx-path>) </idx-path></word>
4. search.query(<word>, "ActiveDoc")</word>

JS API的文档中有说明，方法2和3是有安全限制的，因此，从某个路径（包括UNC路径）加载索引文件的功能在没有提升权限的情况下是无法触发的。

方法1可以在没有提升权限的情况下执行，但是不会解析任何嵌入的搜索索引文件。这可能是一个设计上的决定，不向潜在的恶意索引文件开放Onix search API。

然而，使用带有参数“ActiveDOC”的方法4，Acrobat Reader DC和Acrobat DC首先会将嵌入式索引文件保存到文件夹C:/Users//AppData/LocalLow/Adobe/Acrobat/DC/Search中，然后开始解析它！因此，使用“ActiveDoc”参数，攻击者就有机会

PoC文件PoC.pdf可以用来对描述的行为进行验证。这是一个带有嵌入式搜索索引的PDF文件，下面是从文件中截取的一段JavaScript代码片段，执行了上面提到过的四种se

```
console.show();
app.alert("[i] trying method 1: search.query(\"test\")");
try {
	search.query("test");
	console.println("[+] search.query succeeded without error! you should NOT have hit
onix32!ixCreateIndexManager");
} catch(e) {
	console.println("[??] search.query method 1 failed: " + e.toString());
}

app.alert("[i] trying method 2: search.query(\"test\", \"Index\", \"/somepath.pdx\")");
try {
	search.query("test", "Index", "/somepath.pdx");
	console.println("[??] search.query method 2 succeeded without error!");
} catch(e) {
	console.println("[!] search.query method 2 failed: " + e.toString());
}

app.alert("trying method 3: search.query(\"test\", \"Folder\", \"/somepath.pdx\")");
try {
	search.query("test", "Folder", "/somepath.pdx");
	console.println("[??] search.query method 3 succeeded without error!");
} catch(e) {
	console.println("[!] search.query method 3 failed: " + e.toString());
}

app.alert("[i] trying method 4: search.query(\"test\", \"ActiveDoc\") - now you should hit
onix32!ixCreateIndexManager!");
try {
	search.query("test", "ActiveDoc");
	console.println("[+] search.query method 4 succeeded without error!");
} catch(e) {
	console.println("[??] search.query method 4 failed: " + e.toString());
}
```

先知社区

要想确认成功攻击到Onix解析引擎需要执行以下几个步骤：

1. 删除C:\Users\<user>\AppData\LocalLow\Adobe\Acrobat\DC\Search文件夹下的所有文件，避免加载缓存的索引文件。</user>

2. 打开Acrobat Reader DC
3. 开启调试器并在onix32!ixCreateIndexManager方法下断点。（当开始解析索引文件时，首先会调用这个方法）
4. 打开poc.pdf文件
   a. 在每次search.query方法调用之前，会弹出4次警告
   b. 使用方法1来调用search.query是不会触发onix32!ixCreateIndexManager函数的
   c. 使用方法2和3来调用search.query是不会触发onix32!ixCreateIndexManager函数的（还会抛出异常）
   d.
   使用方法4来调用search.query能够成功触发onix32!ixCreateIndexManager函数，而且，你可以看到在第四次调用search.query方法过程中，，嵌入式索引文件会被写

这个poc代码将强制弹出搜索对话框并在PDF中启动搜索，因此可以证明我们已经触发了文件解析代码。

总结

个人研究可以揭示出如此多的问题是令人十分惊讶的，甚至连供应商也一度以为该攻击面已经被修补了。不管怎样，Adobe最终找到了一种科学的方法来修复这种攻击面上的

下次再见。

你可以关注我的Twitter@AbdHariri,或是关注我们的团队以了解最新的漏洞利用技术和安全补丁。

点击收藏 | 0 关注 | 1
上一篇：在Philips.com中反射型xss 下一篇：Netatalk CVE-2018...
1. 0 条回复
   - 动动手指，沙发就是你的了！

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