隐藏 17 年的 Office 远程代码执行漏洞现 POC 样本

启明星辰提供解决方案

2017 年 11 月 14 日，微软发布了 11 月份的安全补丁更新，其中比较引人关注的莫过于悄然修复了潜伏 17 年之久的 Office 远程代码执行漏洞(CVE-2017-11882)。该漏洞为 Office 内存破坏漏洞，影响目前流行的所有 Office 版本。攻击者可以利用漏洞以当前登录的用户的身份执行任意命令。

由于漏洞影响面较广，漏洞披露后，**金睛安全研究团队**持续对漏洞相关攻击事件进行了关注。11 月 19 日，我们监控到了已有漏洞 POC 在网上流传，迅速对相关样本进行了分析。目前该样本全球仅微软杀毒可以检测。

漏洞影响版本

- Office 365
- Microsoft Office 2000
- Microsoft Office 2003
- Microsoft Office 2007 Service Pack 3
- Microsoft Office 2010 Service Pack 2
- Microsoft Office 2013 Service Pack 1
- Microsoft Office 2016

漏洞事件分析

漏洞出现在模块 EQNEDT32.EXE 中，该模块为公式编辑器，在 Office 的安装过程中被默认安装。该模块以 OLE 技术（Object Linking and Embedding，对象链接与嵌入）将公式嵌入在 Office 文档内。

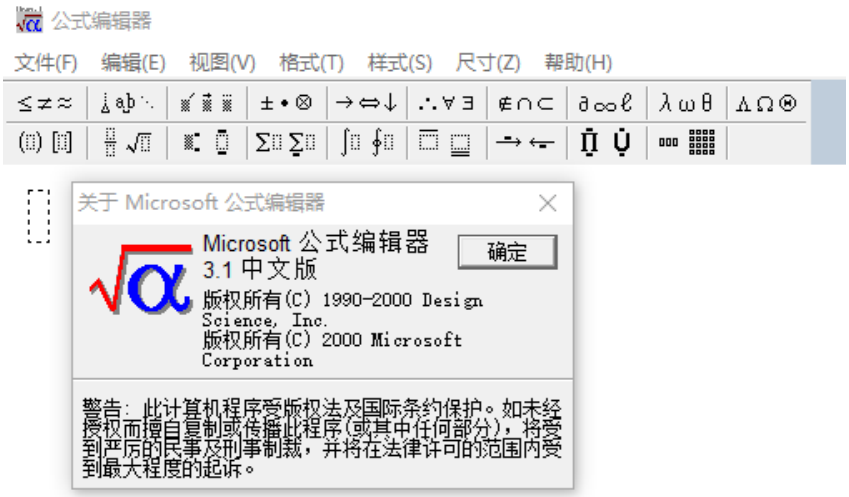


图 1 - Microsoft 公式编辑器

当插入和编辑数学公式时，EQNEDT32.EXE 并不会被作为 Office 进程（如 Word 等）的子进程创建，而是以单独的进程形式存在。这就意味着对于 WINWORD.EXE, EXCEL.EXE 等 Office 进程的保护机制，无法阻止 EQNEDT32.EXE 这个进程被利用。

由于该模块对于输入的公式未作正确的处理，攻击者可以通过刻意构造的数据内容覆盖掉栈上的函数地址，从而劫持程序流程，在登录用户的上下文环境中执行任意命令。

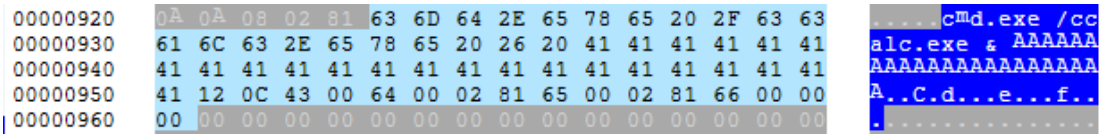


图 2 - CVE-2017-11882 POC 中所执行的命令

国外安全厂商 0patch 对修补前后的相应补丁进行了对比，发现更新补丁后的程序版本是使用汇编方式进行修补的，并据此推测由于年代久远，微软或已丢失相关程序的源代码。再者，当时(2000 年)的编译器所编译的程序并不包含 ASLR 等漏洞缓解措施，因此该模块必将吸引更多黑客对其进行漏洞挖掘。为安全起见，强烈建议用户取消对该模块的注册，解决方案请参考文章结尾部分。

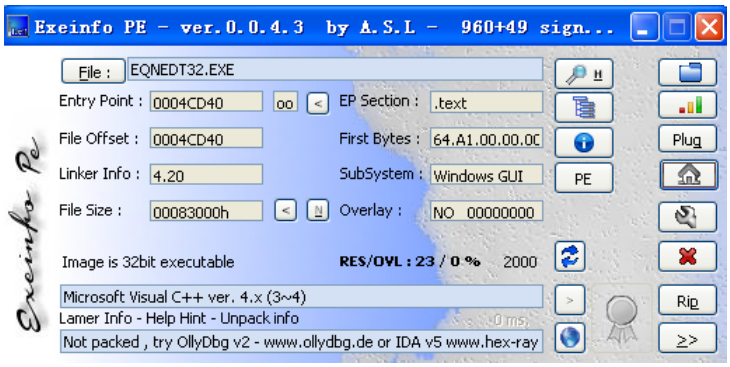


图 3 公式编辑器使用较低版本编译器编写（Visual C++ 3~4 版）

```
ProcessName: EQNEDT32
Source     : Running Process
Id         : 2976

DEP:
    Enable                      : off
    Disable ATL                 : off

ASLR:
    BottomUp                    : off
    ForceRelocate               : off
    HighEntropy                 : off
    DisallowStripped            : off

StrictHandle:
    RaiseExceptionOnInvalid     : off
    HandleExceptionsPermanently : off

System Call:
    DisallowWin32kSysCalls      : off

ExtensionPoint:
    DisableExtensionPoints      : off

DynamicCode:
    ProhibitDynamicCode         : off
    AllowThreadOpt              : off
    AllowRemoteDowngrade        : off

CFG:
    EnableCFG                   : off
    EnableExportSuppression     : off
    StrictMode                  : off

BinarySignature:
    MicrosoftSignedOnly         : off
    StoreSignedOnly             : off
    MitigationOptIn             : off

FontDisable:
    DisableNonSystemFonts       : off
    AuditNonSystemFontLoading   : off

ImageLoad:
    NoRemoteImages              : off
    NoLowMandatoryLabelImages   : off
    PreferSystem32Images        : off
```

图 4 - EQNEDT32.exe 未见任何漏洞环节措施

漏洞分析

（1）基本信息

截止分析报告撰写时，只发现了 RTF 类型的 CVE-2017-11882 漏洞利用文档。触发漏洞的 OLE 对象加载方式与今年上半年的 CVE-2017-0199 漏洞类似，使用了\objupdate 控制字来保证 OLE 对象的自动更新和加载。

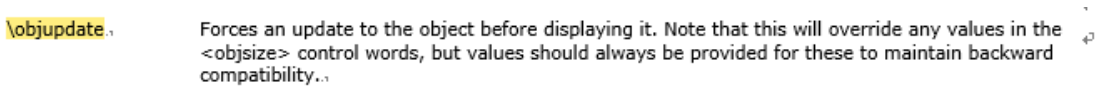


图 5 - RTF 标准文档中对\objupdate 控制字的说明

该 OLE 对象的类型为“Equation.3”，即公式编辑器 3.0 类型对象，该公式对象使用了 CFB

格式进行存储。

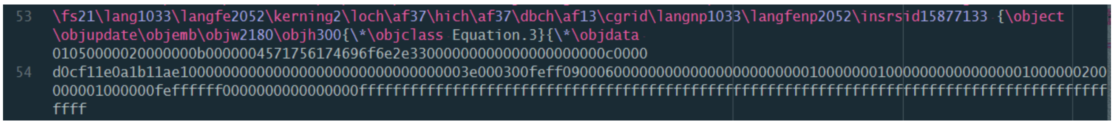


图 6 – 漏洞文档中的 OLE 数据头

对该 OLE 对象进行提取和分析，可以发现公式的相关内容存放在\x01CompObj 流之后。

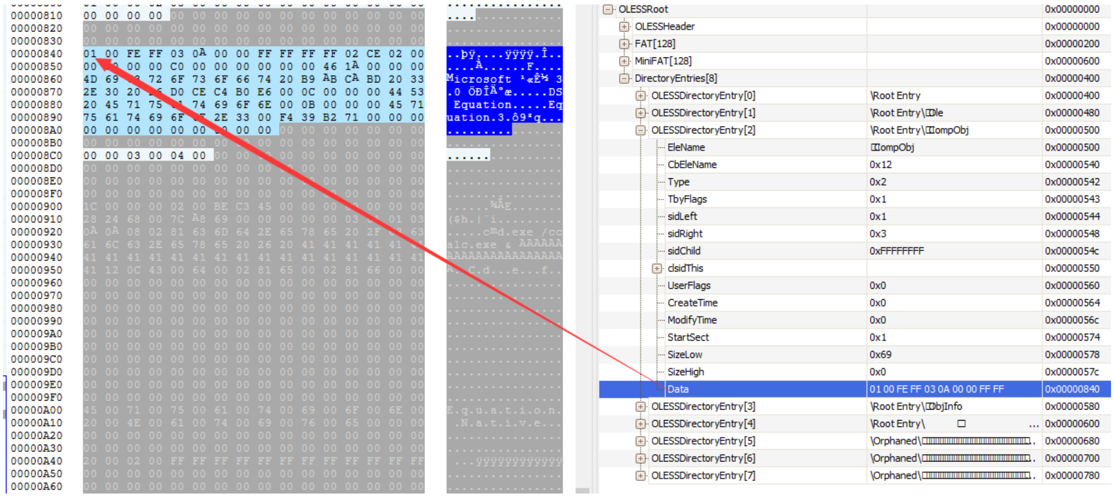


图 7 – 公式对象

（2）详细分析

公式的内容使用了一种名为 MTEF v.3 的二进制格式进行存储。该格式的头部为 28（0x1C）个字节，定义如下：

```
struct EQNOLEFILEHDR {
    WORD    cbHdr;      // 格式头长度，固定为 0x1C。
    DWORD   version;    // 固定为 0x00020000。
    WORD    cf;         // 该公式对象的剪贴板格式。
    DWORD   cbObject;  // MTEF 数据的长度，不包括头部。
    DWORD   reserved1; // 未公开
    DWORD   reserved2; // 未公开
    DWORD   reserved3; // 未公开
    DWORD   reserved4; // 未公开
};
```

在漏洞利用文档中，该结构如下所示。

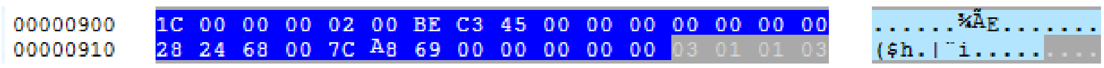


图 8 – 公式头结构

对上图的解析如下表所示。

偏移量	变量名	说明	值
0-1	cbHdr	公式头大小	0x001C
2-5	version	版本号	0x00020000

6-7	cf	剪贴板格式	0xC3BE
8-11	cbObject	MTEF 数据长度	0x45，即 69 字节
12-15	reserved1	未公开	0x00000000
16-19	reserved2	未公开	0x00682428
20-23	reserved3	未公开	0x0069A87C
24-27	reserved4	未公开	0x00000000

紧随该公式头结构的数据为公式数据。公式数据使用字节流进行存储。

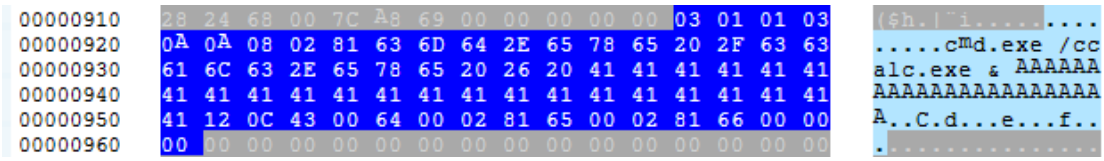


图 9 – 公式数据

MTEF v.3 公式数据中，前 5 个字节为数据头，解析如下表所示。

偏移量	说明	值
0	MTEF 版本号	0x03
1	该数据的生成平台	0x00 表示在 Macintosh 平台生成，0x01 表示在 Windows 平台生成。此处为 0x01。
2	该数据的生成产品	0x00 表示由 MathType 生成，0x01 表示由公式编辑器生成。此处为 0x01。
3	产品主版本号	0x03
4	产品副版本号	0x0A

在数据头之后的字节流即为公式数据。

数据 0x0A 所对应的数据类型为 SIZE，只占用一个字节。

数据 0x08 所对应的数据类型为 FONT，文档中数据的解析如下表所示。

数值	解释
0x08	FONT 记录标志
0x02	typeface 类型
0x81	字体风格
0x636D642E.....	字体名（以空字符结尾），即图 9 中的 cmd.exe...字符串

而问题正出在对字体名的解析上。该公式编辑器只为字体名分配了 36 个字节的空间，而处理时并未对字体名的长度做合理性检验，导致了栈溢出的发生。在图 9 中的一长串"A"之后可以发现一个硬编码的地址 0x00430C12。

```
.text:0041160F dstbuf          = byte ptr -28h
.text:0041160F var_4           = dword ptr -4
.text:0041160F overflowbuf     = dword ptr  8
.text:0041160F arg_4           = dword ptr  0Ch
.text:0041160F arg_8           = dword ptr  10h
.text:0041160F
.text:0041160F                 push    ebp
.text:00411610                 mov     ebp, esp
.text:00411612                 sub     esp, 88h
.text:00411618                 push    ebx
.text:00411619                 push    esi
.text:0041161A                 push    edi
.text:0041161B                 mov     word ptr [ebp+var_4], 0FFFFh
.text:00411621                 mov     word ptr [ebp+var_38], 0FFFFh
.text:00411627                 mov     edi, [ebp+overflowbuf]
.text:0041162A                 mov     ecx, 0FFFFFFFFh
.text:0041162F                 sub     eax, eax
.text:00411631                 repne scasb
.text:00411633                 not     ecx
.text:00411635                 lea     eax, [ecx-1]
.text:00411638                 mov     [ebp+var_34], ax
.text:0041163C                 mov     edi, [ebp+overflowbuf]
.text:0041163F                 mov     ecx, 0FFFFFFFFh
.text:00411644                 sub     eax, eax
.text:00411646                 repne scasb
.text:00411648                 not     ecx
.text:0041164A                 sub     edi, ecx
.text:0041164C                 mov     eax, ecx
.text:0041164E                 mov     edx, edi
.text:00411650                 lea     edi, [ebp+dstbuf] ; 栈上大小36字节的空间
.text:00411653                 mov     esi, edx
.text:00411655                 shr     ecx, 2
.text:00411658                 rep movsd               ; 未经校验，直接拷贝，造成栈溢出
.text:0041165A                 mov     ecx, eax
.text:0041165C                 and     ecx, 3
.text:0041165F                 rep movsb
.text:00411661                 lea     eax, [ebp+dstbuf]
.text:00411664                 push    eax             ; lpSrcStr
.text:00411665                 call    sub_451DE0
.text:0041166A                 add     esp, 4
.text:0041166D                 call    sub_420FA0
.text:00411672                 mov     [ebp+var_2C], ax
```

图 10 - 漏洞代码

通过对下面的两张图进行对比，可以明白栈溢出的触发过程。

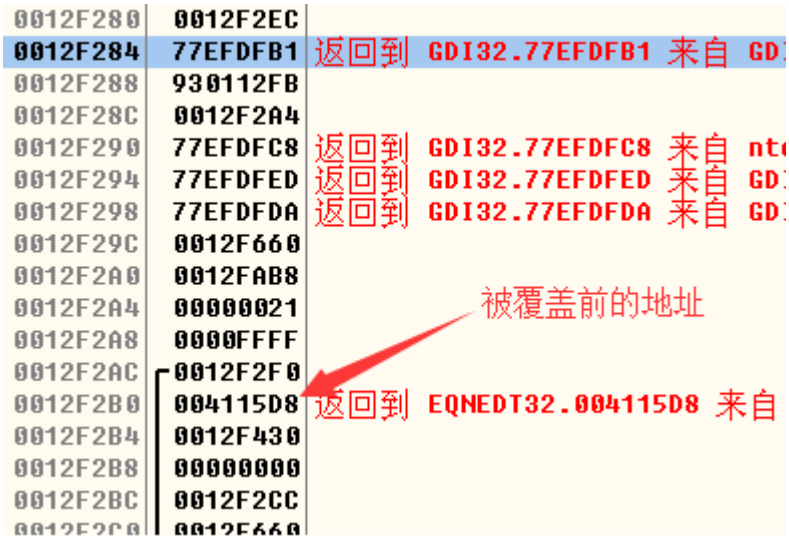


图 11 - 被覆盖前的栈数据

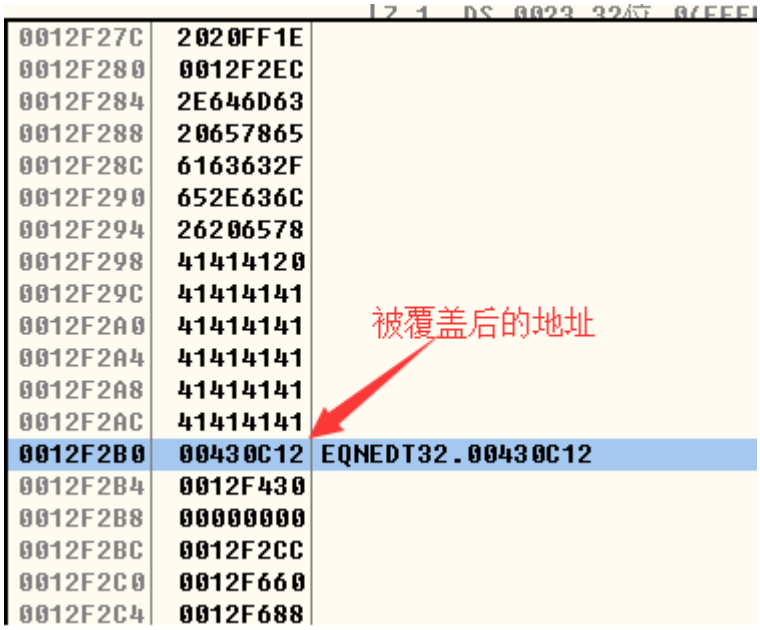
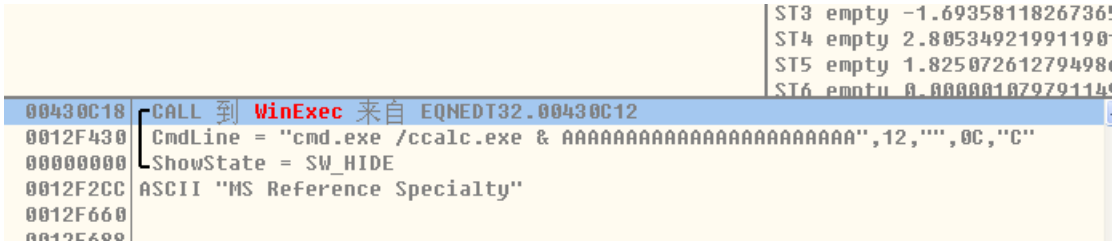


图 12 - 被覆盖后的栈数据

被修改后的函数调用如下图所示，在前文中已经提到，该公式编辑器并没有开启 ASLR。这个硬编码的地址 0x00430C12 对应于对函数 WinExec 的调用。因而该字体名对应的命令得以执行。



解决方案

（1） 微软已经对此漏洞做出了修复。

- 下载 https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2017-11882 更新补丁进行修补
- 开启 Windows Update 功能，定期对系统进行自动更新

（2） 由于该公式编辑器已经 17 年未做更新，可能存在大量安全漏洞，建议在注册表中取消该模块的注册。

- 按下 Win+R 组合键，打开 cmd.exe
- 输入以下两条命令：

```
reg add “HKLM\SOFTWARE\Microsoft\Office\Common\COM Compatibility\{0002CE02-0000-0000-C000-000000000046}” /v “Compatibility Flags” /t REG_DWORD /d 0x400

reg add “HKLM\SOFTWARE\Wow6432Node\Microsoft\Office\Common\COM Compatibility\{0002CE02-0000-0000-C000-000000000046}” /v “Compatibility Flags” /t REG_DWO
```

```
RD /d 0x400
```

（3） 天阗高级持续性威胁检测与管理系统的流行威胁库已经可以检测该漏洞的利用样本，请及时升级。