“狼毒草”APT 组织—持续多年的定向攻击威胁

启明星辰公司-金睛安全研究团队

2018 年 8 月 15 日

北京启明星辰信息安全技术有限公司

Beijing Venus Information Security Tech., Inc.

目 录

背景 2
一、攻击事件还原 2
1.1 攻击过程分析 2
1.2 载荷分析 2
二、攻击样本分析 3
2.1 主 Dropper 分析 3
2.2 释放文件分析 8
2.2.1 Loader 和反 AV 模块分析 8
2.2.2 Downloader 分析 11
2.2.3 后门模块分析 16
2.2.4 插件分析 17
三、关联样本分析 19
3.1 配置信息比较 20
3.2 释放文件比较 21
四、溯源分析 23
五、总结 25
六、IOC 25
七、参考 27

背景

2018 年 7 月，启明星辰金睛安全研究团队捕获到了一类新的定向攻击样本，经过分析确认为之前未知的 APT 组织攻击样本。经过分析，我们追溯到该组织最早于 2013 年就开始活动。我们将该 APT 组织命名为“狼毒草”。

下面我们将从最新发现的攻击事件所使用的手段以及相关的恶意样本入手，同时对我们关联到的历史攻击样本进行关联分析，以便更好地勾勒出该 APT 组织相貌。

一、攻击事件还原

1.1 攻击过程分析

此次攻击中，我们获得到的样本来源于受害者的内网服务器，此内网服务器为受害者用于部署内部办公系统的服务器。可以推断出攻击者借助内网服务器做跳板对其他受害者进行水坑攻击以达到横向传播的目的。

1.2 载荷分析

在此次攻击中，用于水坑攻击的攻击载荷为一个名为 flash_security_component_installer_1.0.0.2.rar 的压缩包。

压缩包内含有两个文件，其中一个为安装说明文档，另一个为伪装成 flash

安装包的恶意程序。

flash_security_component_installer_1.0.0.2.exe	220.0 KB	139.0 KB	应用程序	2018-07-21 16:06
flash安全组件安装说明.doc	93.7 KB	88.4 KB	DOC 文档	2018-07-21 16:48

从获取到的攻击样本的文件名，以及文档内容的拼写习惯可以初步确认，攻击者对中文非常熟悉，并且精通社会工程学。下面将具体对伪装成 flash 安装包的恶意软件进行详细分析。

二、攻击样本分析

2.1 主 Dropper 分析

伪装成 flash 安装包的程序具有较多功能，其会根据系统版本、当进程权限、杀软等情况，执行不同的释放流程。

1、程序执行后，首先检测杀软 ESET 和小红伞，有任意一个立即退出进程。

通过枚举控制面板卸载信息来检测 ESET 是否存在。

```
地址      值       注释
0012F830  004074D5 CALL 到 strstr 来自 flash_se.004074CF
0012F834  0012FB54  s1 = "IDA Pro v6.8 and Hex-Rays Decompiler (ARM,x64,x86)"
0012F838  0012FEEC  s2 = "ESET"
```

通过查找进程 avira.Systray.exe 来检测小红伞是否存在。

```
地址      值       注释
0012ED24  0012ED74 UNICODE "winlogon.exe"
0012ED28  0012ED8C
0012ED2C  003A8152 UNICODE "avira.Systray.exe"
```

2、若不存在，则会根据系统版本释放多个文件，并分为以下情况：

（1）对于 Vista 以后系统，若当前进程没有管理员权限，由于 UAC 的限制，该文件无法被直接复制到 system32 目录下，此时会使用 CIA Vault 7 里绕过 UAC 的办法。

创建进程 wusa.exe，获取其令牌复制给自身进程。随后调用

CreateProcessWithLogon 创建新的自身进程，username 传入"uac"，domain 传入"is"，password 传入"useless"。

```
004044DD        mov     word ptr [ebp+Username], 'u'
004044E6        push    eax             ; lpProcessInformation
004044E7        lea     eax, [ebp+StartupInfo]
004044ED        push    eax             ; lpStartupInfo
004044EE        push    ebx             ; lpCurrentDirectory
004044EF        push    ebx             ; lpEnvironment
004044F0        push    ebx             ; dwCreationFlags
004044F1        push    ebx             ; lpCommandLine
004044F2        lea     eax, [ebp+Username]
004044F8        push    [ebp+lpApplicationName] ; lpApplicationName
004044FB        mov     word ptr [ebp+Username+2], 'a'
00404504        mov     word ptr [ebp+Username+4], 'c'
0040450D        mov     word ptr [ebp+Username+6], bx
00404514        push    LOGON_NETCREDENTIALS_ONLY ; dwLogonFlags
00404516        push    offset Password ; "useless"
0040451B        push    offset Domain   ; "is"
00404520        push    eax             ; lpUsername
00404521        call    ds:CreateProcessWithLogonW
```

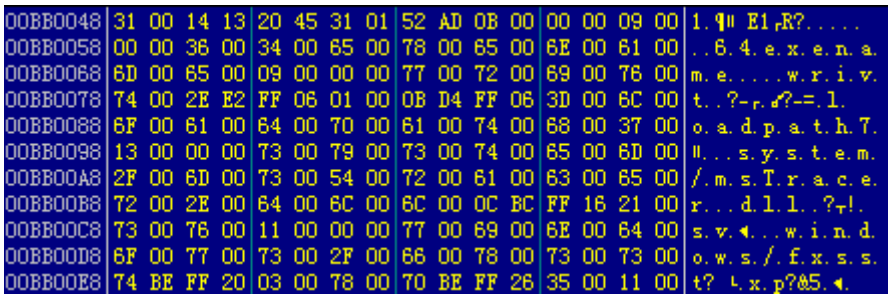
并且，在 Vista 之后的系统，在存在管理员权限时，会根据杀软情况，来停止 WSearch 服务（SearchIndexer.exe,SearchProtocolHost.exe）。只有存在 360tray.exe(360)、ekrn.exe(ESET)、 kxetray.exe(江民)中的任意一个时，才停止 WSearch 服务。如果都不存在，则直接执行后续的释放。特别是如果存在 360tray.exe，还会把 WSearch 服务设置为自动启动。

（2）Vista 之前系统不存在上述行为，会直接执行释放文件。

3、下面为释放文件的主要流程。

（1）首先会加载图片资源，并进行解密

```
004012F1        push    RT_BITMAP
004012F3        push    110
004012F5        push    eax
004012F6        lea     eax, [ebp+var_194]
004012FC        push    eax
004012FD        call    loadcfqfromres
```



（2）资源开头部分是一些路径的配置数据，后半部分是加密的 PE 数据，包括文件名分别为 temp0、fake、acess、wrivt.exe 的程序，wrivt.exe 在 Vista 之后的 64 位系统上会释放出来。

根据系统版本情况，这几个文件的释放路径有所不同。

配置项	数据	描述
64exename	wrivt.exe	64 位下删除自身的模块
64loadpath7	system/msTracer.dll	64 位 Win7 之后系统下的 temp0 释放路径
64loadpathsv	windows/fxsst.dll	64 位 windows server 系列系统下的 temp0 释放路径
64loadpathxp	windows/fxsst.dll	64 位 Vista 之前系统下的 temp0 释放路径
loadpath7	system/msTracer.dll	32 位 Win7 之后系统下的 temp0 释放路径
loadpathsv	windows/fxsst.dll	32 位 windows server 系列系统下的 temp0 释放路径
loadpathxp	system/srvlic.dll	32 位 Vista 之前系统下的 temp0 释放路径
mainpath	commonappdata/Intel/Runtime/fake	fake 的释放路径
vfpath	commonappdata/Intel/Runtime/acess	acess 释放路径

启明星辰公司 版权所有

AfterInstallation	RemoveInstaller	释放后删除自身即假冒的 flash 安装包

（3）根据系统是 32 位还是 64 位，分别加载资源里对应的 temp0、fake、acess 的 32 位数据或 64 数据，并释放到所配置的路径里。

（4）需要特别指出的是，对应 Vista 之前系统，直接调用 MoveFileExW 把 temp0 拷贝到系统目录。但是对 Vista 之后系统，根据杀软情况以及 32 位还是 64 位系统，选择不同方式来拷贝 temp0 到系统目录。

对于 32 位系统，且 360tray.exe、rstray.exe、qqpctray.exe 进程都不存在，则通过执行

```
makecab.exe /V1 "C:\Users\<username>\AppData\Local\Temp\msTracer.dll" "C:\Users\<username>\AppData\Local\Temp\msTracer.dll.msu"
```

将 msTracer.dll 文件封装成一个 msu(Windows 更新)文件，并通过 wusa.exe（windows 更新文件封装程序）执行，把 msTracer.dll.msu 即 temp0 安装到系统目录。

```
wusa.exe /quiet "C:\Users\<username>\AppData\Local\Temp\\msTracer.dll.msu" /extract:C:\Windows\system32
```

若存在 360tray.exe、rstray.exe、qqpctray.exe 中的任意一个进程，则使用 Com Elevation Moniker 技术绕过 UAC，再进行拷贝。绕过 UAC 的主要步骤是：

向 HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Uninstall 添加一个 EsayNote 项，在其 UninstallString 里设置为

C:\Users\<username>\AppData\Local\Temp\xcopy.exe "C:\Users\<username>\AppData\Local\Temp\msTracer.dll" "C:\Windows\system32\"

```
0040C0E4 ; const WCHAR pszName
0040C0E4 pszName:                                ; DATA XREF: sub_4011F6+48↑o
0040C0E4                 unicode 0, <Elevation:Administrator!new:{FCC74B77-EC3E-4DD8-A80B-008A>
0040C0E4                 unicode 0, <702075A9}>,0
```

然后调用 COM 组件 IARPUninstallStringLauncher 的方法执行，最终会执行到 UninstallString 的指令即 xcopy.exe msTracer.dll 到 system32 目录下。[1][2]

而对于 64 位系统，首先检测 360tray.exe 进程，如果不存在，继续检测 rstray.exe、qqpctray.exe，如果也都不存在，则执行上述的绕过 UAC 的方法拷贝。如果 rstray.exe、qqpctray.exe 存在任意一个，则通过上述的安装包+wusa.exe 方式拷贝。

当存在 360tray.exe 进程时，会向临时目录释放 wrivt.exe 并运行。wrivt.exe 唯一的功能就是删除之前释放的 temp0。换句话说，在 Vista 之后的 64 位系统，如果安装了 360，样本则直接放弃执行后续动作。

（5）fake、access 加密后释放在 commonappdata/Intel/Runtime 目录下。

（6）上述文件释放完成后，检测配置里的 AfterInstallation，如果是 RemoveInstaller，则尝试删除自身。同样会先检测杀软进程，如果 KVSrvXP.exe、twssrv.exe、nis.exe 存在任意一个，则使用延迟删除技术来删除自身。

如果三个杀软进程都不存在，则释放批处理文件 C:\Documents and Settings\All Users\Application Data\Revinst\upvent.bat 来删除自身。

```
:nf
del "C:\Users\Administrator\Desktop\flash_security_component_installer_1.0.0.2.exe"
if exist "C:\Users\Administrator\Desktop\flash_security_component_installer_1.0.0.2.exe" goto nf
del "%~f0"
```

（7）最后对于 Vista 以后系统，会尝试启动搜索服务 WSearch。根据杀软情况，有两种方式。如果 360tray.exe、ekrn.exe、kxetray.exe 都不存在，直接创建进程 SearchProtocolHost.exe。

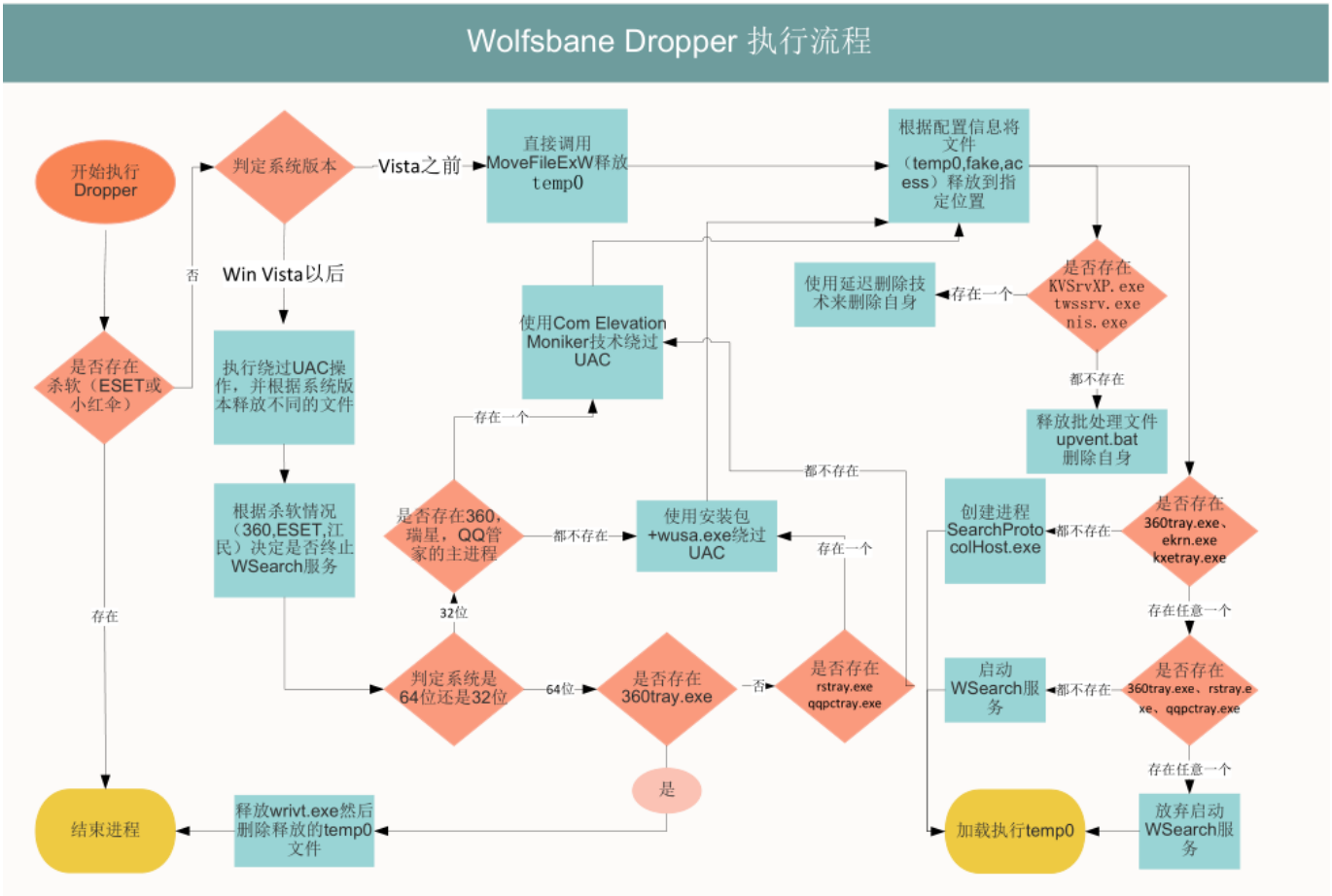
```
地址      值       注释
0012EC54  00404A58 CALL 到 WinExec 来自 flash_se.00404A56
0012EC58  00C25009 CmdLine = "cmd.exe /c C:\WINDOWS\system32\SearchProtocolHost.exe"
0012EC5C  00000000 ShowState = SW_HIDE
```

[1] https://gist.github.com/Cr4sh/9b5b7f663ff0b5b798f3221ea91c5e81

[2] https://3gstudent.github.io/3gstudent.github.io/通过 COM 组件 IARPUninstallStringLauncher 绕过 UAC/

如果上述三个杀软存在任意一个，则继续检测下列三个杀软 360tray.exe、rstray.exe、qqpctray.exe，如果存在任意一个，则放弃启动 WSearch 服务，退出进程。如果都不存在，查询 WSearch 服务状态，不是启动状态的话，则立即启动。

至此，我们总结出 Dropper 执行流程，见下图。



2.2 释放文件分析

以下为释放的文件对应功能，下面将针对每个释放文件进行分析。

文件名	功能
temp0	Loader 模块
fake	下载并执行插件模块
acess	对抗金山毒霸的模块

2.2.1 Loader 和反 AV 模块分析

上节我们对主 Dropper 根据系统版本、杀软等情况，执行不同的释放流程进行了分析与总结。其中 temp0 作为 Loader 模块，释放的路径以及命名尤为重要。下面我们将针对 temp0 的加载方式以及 temp0 加载的反 AV 模块 acess 进行详细分析。

1、在这里 Dropper 使用了 DLL 劫持技术对 temp0 进行加载。

（1）在 32 位 XP 系统下，temp0 被拷贝为 system/srvlic.dll，此时将利用 LanmanServer 服务进行加载。LanmanServer 的 svchost.exe 进程会加载 srvsvc.dll 文件，而 srvsvc.dll 文件会在执行 LoadLicensingLibrary 这一 API 时加载 srvlic.dll 文件。

```
74FF755E                mov     edi, offset aSrvlic_dll ; "srvlic.dll"
74FF7563                push    edi             ; wchar_t *
74FF7564                mov     esi, eax
74FF7566                call    ds:wcslen
```

（2）在 32 位 Window 7 系统下，temp0 被拷贝为 system/msTracer.dll。

此时利用的是搜索服务 WSearch 的进程 SearchProtocolHost.exe 对 msTracer.dll 进行加载。

```
push    offset aMstracer_dll ; "msTracer.dll"
push    0Ch             ; int
lea     ecx, [ebp+lpFilename]
call    sub_100D135
push    [ebp+lpFilename] ; lpLibFileName
call    esi ; LoadLibraryW
```

（3）其它系统环境下，temp0 的加载很类似，不再详细说明。

2、temp0 的主要功能是读取此前释放在 commonappdata/Intel/Runtime/下的 fake、acess 模块，并在内存中执行。但前提必须满足 3 个条件中的其中一个：当前用户为 System 用户，当前进程是 explorer.exe，当前进程是 SearchProtocolHost.exe。

3、当条件满足后，temp0 会加载图片资源，按字节异或，初始密钥为 0x4E，密钥每异或一次递增 0x0F，直至解密完成为止，最终会解出 fake 和 acess 的路径。

4、紧接着，temp0 会先解密 acess（对抗金山毒霸的模块），并在内存里执行 needmid 和 setmid 函数。根据 needmid 和 setmid 的返回结果来决定是否继续加载 fake 模块。

启明星辰公司 版权所有

5、事实上，如果 needmid 函数发现系统上没有安装金山毒霸，或者虽然安装了金山毒霸，但安装目录里有 version.dll，temp0 会继续加载 fake 模块，后面将会详细说明这一点。

6、acess 首先会通过枚举注册表路径 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\xx 下的 DisplayName 来检测是否存在金山毒霸。

7、此外，其还会检查毒霸安装目录下是否存在 version.dll，存在则直接返回。然后 temp0 会继续加载 fake 模块。如果不存在，即尝试把 acess 保存到毒霸的安装目录下的 version.dll。为了实现这一步，acess 使用了一种 dll 注入方法，即基于 MS 应用程序验证机制的 dll 注入，也被称为 DoubleAgent。

其原理和利用流程如下：

应用程序验证是微软提供的一种机制，在某个 exe 的测试阶段，为其注册一个 VeriferDlls。VerfierDlls 在 exe 启动时被加载，通过 API Hook 的形式，实时检验 exe 的各种隐含错误或漏洞。

而启用应用程序验证机制，需要为 exe 创建两个映像劫持下的注册表项。一是创建 DWORD 类型的 GlobalFlag，并指定其值为 0x100。二是创建字符串项 VerifierDlls，并指定 Dll 的名称。

在此，acess 把自身拷贝到系统目录并命名为 vfpod.dll，然后对毒霸卸载程序 uni0nst.exe 的注册 VerfierDlls。

通过查看注册表，可以看到其内已经注册了 VerifierDlls，并指定了 dll 名称。“HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\uni0nst.exe”

名称	类型	数据
(默认)	REG_SZ	(数值未设置)
GlobalFlag	REG_SZ	0x100
VerifierDlls	REG_SZ	vfpod.dll
VerifierFlags	REG_DWORD	0x80000000 (2147483648)

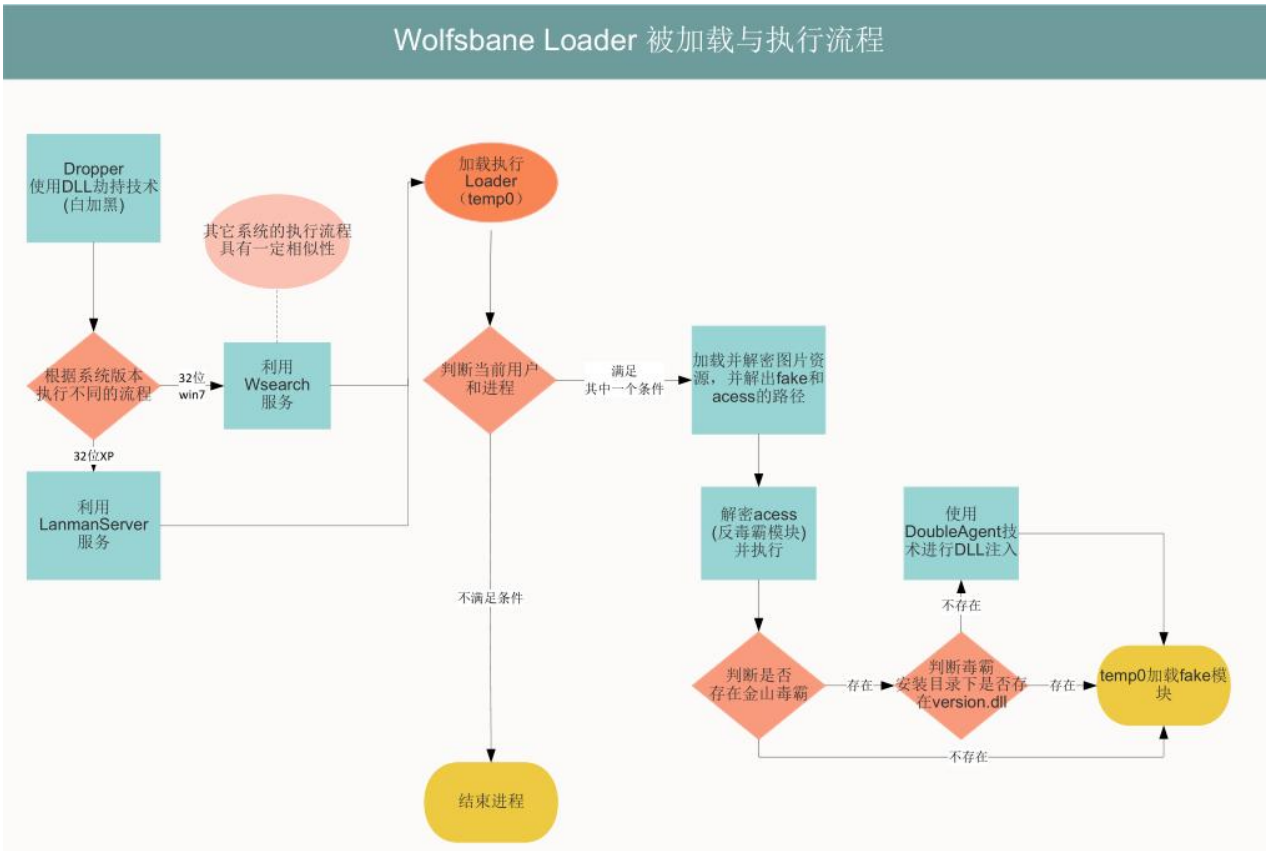
随后运行 uni0nst.exe，而 vfpod.dll 即 acess 会被 uni0nst.exe 加载运行。

```
地址      值        注释
0012F284  009C3107  CALL 到 WinExec 来自 009C3101
0012F288  0012F4A8  CmdLine = "c:\program files\kingsoft\kingsoft antivirus\\uniOnst.exe"
0012F28C  00000000  ShowState = SW_HIDE
```

此时 vfpod.dll 会将自身拷贝到金山毒霸安装目录下的 version.dll。

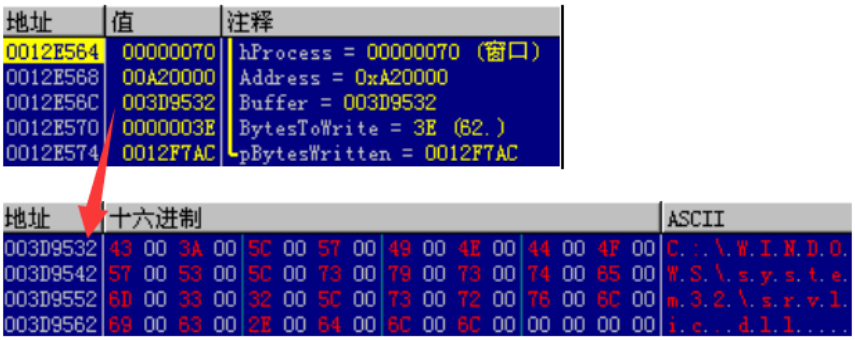
若金山毒霸进程加载 version.dll，则会挂钩 OpenProcess 函数，当金山毒霸打开 temp0 和 fake 栖身的宿主进程 SearchIndexer.exe、explorer.exe、searchprotocolhost.exe 时，则会返回失败，temp0 和 fake 即可免于被扫描。这也是为什么发现安装目录里有 version.dll 时，temp0 会大胆地继续加载 fake 模块的原因。

同样，我们也将上述流程整理成图，如下所示。



2.2.2 Downloader 分析

1、fake 是下载模块，主要起下载其它插件的作用。对于 Vista 之前系统，fake 模块的最终宿主进程是 explorer.exe，如果此时不是运行在此进程，其会把 temp0 重新注入到 explorer.exe。



2、对于 Vista 之后系统，若当前不是 System 用户，也会转到注入 temp0 到 Explorer.exe 的流程。而如果是 System 用户且配置里 WorkMode 是 searchindex 时，会尝试删除防火墙针对 SearchIndexer.exe 的阻断规则。如果删除规则成功，即转到注入 temp0 到 Explorer.exe 的流程。若删除失败，则在当前进程执行下载等功能。

这逻辑显然不符合情理，很可能代码不小心写错了。合理地猜测作者的本意，temp0 被 SearchIndexer.exe 加载，fake 也因此被加载起来。fake 要联网下载插件，故删除阻断规则。删除成功了，显然可以在当前进程 SearchIndexer.exe 执行下载功能。删除失败，才需要重新注入 temp0 到 Explorer.exe。

3、fake 不会主动向 C&C 发送数据，只是被动等待 C&C 发送来的数据，且通信数据经过了加密。通信格式如下：

C&C 发送给 fake 的命令格式：

Size(4)	SecretKey(4)	Magic(4)	OPcode1(4)	OPcode2(4)	PayLoad

Size 等于总长度减去 4，Magic 、OPcode1、OPcode2、PayLoad 都使用 SecretKey 加密。SecretKey 随机生成。

fake 给 C&C 的回应格式：

Size(4)	SecretKey(4)	Magic(4)	PayLoad

Size 等于总长度减去 4，Magic、PayLoad 使用 SecretKey 加密了，SecretKey 是随机生成。

以 OPcode1==0x22836D73、OPcode2==0x6F42E3C0 为例，此命令是获取系统位数即 32 位还是 64 位。

```
Data: 2000000006b1f21793f6e132228d5abfb50728ac9b89
[Length: 36]
00 0c 29 3c a6 24 00 50  56 e7 13 5b 08 00 45 00
00 4c 00 21 00 00 80 06  ec 5d 3c a9 01 56 c0 a8
4f 86 06 1a 04 1f 7e 60  52 99 a5 43 f2 fe 50 18
fa f0 f3 1c 00 00 20 00  00 00 06 b1 f2 17 93 f6
e1 32 22 8d 5a bf b5 07  28 ac 9b 89 89 89 89 89
89 89 b7 b7 b7 b7 b7 b7  b7 b7
```

命令数据总长度是 36 字节，Size 等于 0x20 即 36-4=32。SecretKey 是 0x17f2b106，随后是加密的 Magic 、OPcode1、OPcode2、PayLoad。

利用 SecretKey(0x17f2b106)解密 Magic 、OPcode1、OPcode2、PayLoad：

```
地址      十六进制                                          ASCII
00B70020  06 B1 F2 17 6B 5A 31 C7 73 6D 83 22 C0 E3 42 6F  -彬kZ1玺m?楞Bo
00B70030  FC FF FF FF FF FF FF FF 00 00 00 00 00 00 00 00  ?
```

Magic：0xC7315A6B，OPcode1：0x22836D73，OPcode2==0x6F42E3C0

PayLoad：FC FF FF FF FF FF FF FF，对于此命令无意义。

```
10003147  > 3D 736D8322    CMP   EAX,0x22836D73          loc_10003147
1000314C  .v 75 08         JNZ   SHORT <loc_10003156>
1000314E  .  81F9 C0E3426F CMP   ECX,0x6F42E3C0
10003154  .v 74 58         JE    SHORT <ExecCmd_GetPlatformBits>
10003156  > 3D D2BF5432    CMP   EAX,0x3254BFD2          loc_10003156
1000315B  .v 75 08         JNZ   SHORT <loc_10003165>
1000315D  .  81F9 1797F36F CMP   ECX,0x6FF39717
10003163  .v 74 3C         JE    SHORT <ExecCmd_LoadLibrary>
10003165  > 3D FF9F74B4    CMP   EAX,0xB4749FFF          loc_10003165
1000316A  .v 75 08         JNZ   SHORT <loc_10003174>
1000316C  .  81F9 829710A7 CMP   ECX,0xA7109782
10003172  .v 74 20         JE    SHORT <loc_10003194>
10003174  > 3D 3E30D7DD    CMP   EAX,0xDDD7303E          loc_10003174
```

比较 OPcode1、OPcode2，执行对应的功能。

```
push    offset aSuccess ; "success"
push    [ebp+arg_4]     ; int
call    sub_10002444
or      [ebp+Src], 0FFFFFFFFh
or      [ebp+var_4], 0FFFFFFFFh
mov     esi, eax
lea     eax, [ebp+Src]
push    8               ; Size
push    eax             ; Src
push    esi             ; Dst
call    memcpy
add     esi, 8
push    offset aX86     ; "X86"
push    esi             ; int
call    sub_10002444
```

地址	十六进制	UNICODE
00B7002C	07 00 00 00 73 00 75 00 63 00 63 00 65 00 73 00	•.succes
00B7003C	73 00 FF FF FF FF FF FF FF FF 03 00 00 00 58 00	s.....ㄴX
00B7004C	38 00 36 00 00 00 00 00 00 00 00 00 00 00 00 00	86......

对应的功能函数 ExecCmfd_GetPlatformBits 里把 success 和 X86 字符串拼接起来，中间用 8 个 FF 分隔开来。

```
lea     rdx, aSuccess   ; "success"
mov     rcx, rdi
call    sub_100027B4
lea     rdx, [rsp+38h+Src] ; Src
mov     r8d, 8          ; Size
mov     rcx, rax        ; Dst
mov     rbx, rax
mov     [rsp+38h+Src], 0FFFFFFFFFFFFFFFFh
call    memcpy
lea     rcx, [rbx+8]
lea     rdx, aX64       ; "X64"
call    sub_100027B4
```

如果是 64 位系统，Dropper 释放的 64 位 fake 会拼接 success 和 X64。事实上，C&C 正是依据此返回结果选择下发 32 位还是 64 位的插件。

随后生成随机的 SecretKey，本例是 0x5E2A5642，并对 Magic(0xC7315A6B)和拼接的字符串进行加密。需要指出的是，Magic(0xC7315A6B)是硬编码，并不是直接使用 C&C 命令里的 0xC7315A6B。

启明星辰公司 版权所有

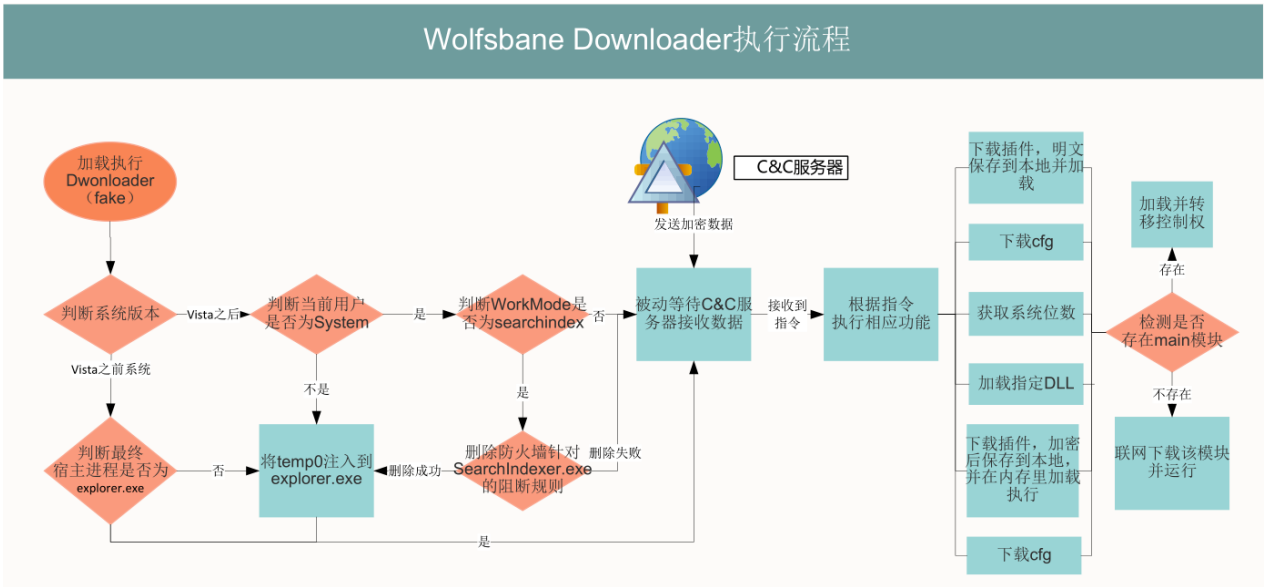
地址	十六进制												ASCII
00B70020	30	00	00	00	42	56	2A	5E	3C	45	EF	4F	13	BB	BB	BB	0...BV*^<E睦O换
00B70030	0C	BB	FA	BB	65	BB	65	BB	53	BB	0C	BB	0C	BB	B8	B8	.机篷篷篮??桓
00B70040	B8	B8	B8	B8	B8	B8	A4	BB	BB	BB	F9	BB	F6	BB	8A	BB	父父父ㄓ换 龌娀
00B70050	BB	BB	BB	BB	00	00	00	00	00	00	00	00	00	00	00	00	换换..........

4、fake 支持以下 6 种命令：

OPcode1	OPcode2	功能
0134A6D30	0100B4627	下载插件，明文保存到本地并加载
0C558B012	05047A6F4	下载 cfg
022836D73	06F42E3C0	获取系统平台即 X86 or X64
03254BFD2	0 6FF39717	加载指定 dll
0B4749FFF	0A7109782	下载插件，加密后保存到本地，并在内存里加载执行
0DDD7303E	0CDF2E7F4	下载 cfg

5、在最终宿主进程里，fake 会检测 main 模块，存在便加载，并把控制权交给 main，至此 fake 任务完成。如果不存在，则联网下载并加载运行。

下图为 Donwloader 的执行流程。



2.2.3 后门模块分析

main 是核心的后门模块，负责与 C&C 通讯，执行 C&C 发来的命令，如下载插件。其维护了一个链表，把自身及它插件的各种功能函数保存在链表里，接收到 C&C 命令后，会去链表里找到对应函数并执行。

```
10089054 aServicecontext:                        ; DATA XREF: sub_100
10089054                 unicode 0, <ServiceContext::ServiceTable>,0
1008908E                 align 10h
10089090 aServiceFunctio:                        ; DATA XREF: sub_100
10089090                                         ; sub_10028580+31↑o
10089090                 unicode 0, <Service::FunctionTableSection>,0
```

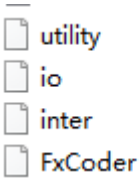
main 自身支持如下的命令：

GetOperationBasicInformation
LoadDll
LoadPlugin
UnloadPlugin
ForkOperation
FindRequiredLibrary
ClientBasicInformation
SettingsService
UpdateSettingsService
UpdateService
UninstallService
ExecuteService
ExecuteServiceAsUser
KeepAliveService
RawModeService
RawModeUDPService
LoadRequiredLibrary

到目前为止，只收到了 GetOperationBasicInformation、ClientBasicInformation 命令，分别为获取系统信息(如 X86 还是 X64)和样本自身信息(如插件版本等)。

2.2.4 插件分析

以下为样本下载的插件列表，下面分别对其作用进行说明。



1、插件 FxCoder：

FxCoder 实现了 Deflate 算法，对数据进行压缩和解压。FxCoder 被下载并加载之后，main 和 C&C 通信通信数据会先进行压缩，然后再用 fake 使用的加密算法进行加密。

2、插件 utility：

utility 是一个功能较复杂的插件，主要进行文件操作，包括文件读写删除，以及对文件建立索引。

涉及的文件类 API 如下：

CreateFileService
ClientStatusService
BasicInformationService
ListFileService
ListFileRecursiveService
OpenFileService
CloseFileService
DeleteFileService
FileSizeService

SeekFileService
WriteFileService
ReadFileService
MoveFileService
WriteFileBytesService
ReadFileBytesService
StartIndexService
QueryIndexService
GetIndexService
GetActiveUsersService
StartIndexToFileService
IndexToFileServicePath

utility 还有 3 个和文件无关的命令，其中 BasicInformationService 是获取 CPU、系统盘符以它们的存储空间大小等。ClientStatusService 是获取系统信息，包括系统版本、机器名称、用户账号、是否开启 UAC、是否管理员权限、IP 地址等。

GetActiveUsersService 获取当前会话的用户名称。

3、插件 inter：

inter 疑似为攻击者提供了额外的预留接口，在 main 插件之外实现了一组新的命令。具体支持如下 5 个命令：

RegisterInterStub
RegisterInjectStub
RegisterInterCondition
ReserveInterOperation
FindProcessIdAndUsers

其中 RegisterInjectStub 是注入模块到指定进程。FindProcessIdAndUsers 为查找进程 ID 和用户。

inter 有一个明文的 dll 资源，会被注入到指定进程。该 dll 提供了 4 个命令：

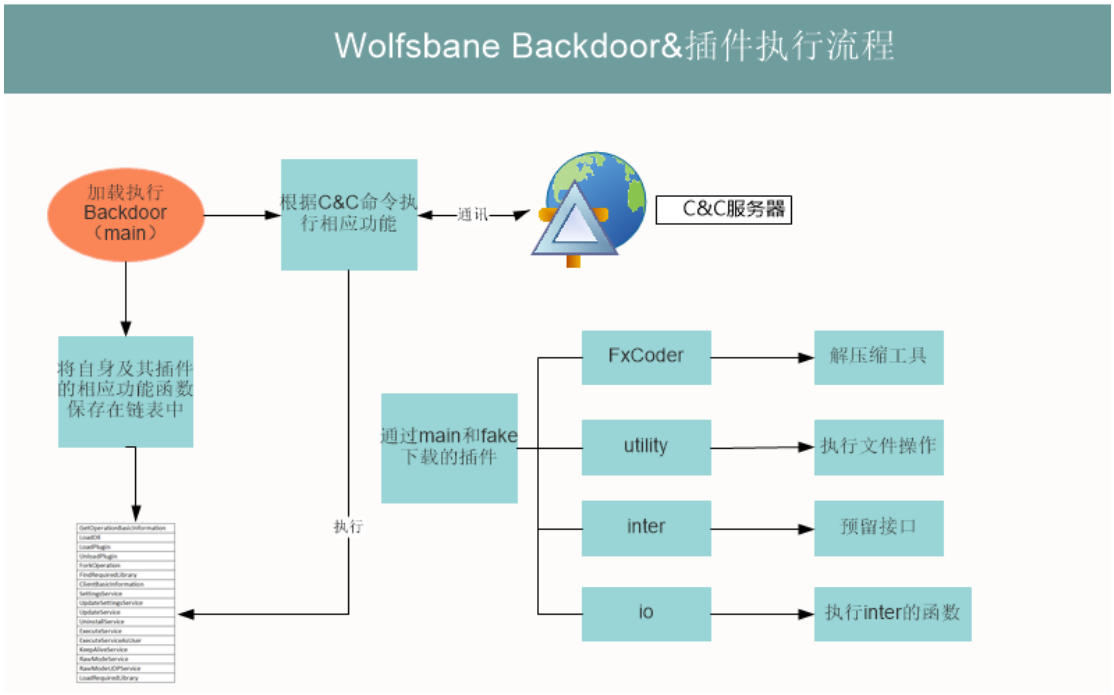
Update
LoadStub
EndInterCommService
ForkOperation2

其中 LoadStub 是用来在内存里加载执行 dll，Update 为执行更新操作。

4、插件 io：

io 会查找 inter 的 RegisterInterStub、RegisterInterCondition 函数并执行。io 有两个 dll 资源，DebugStub 和 DebugDumper。

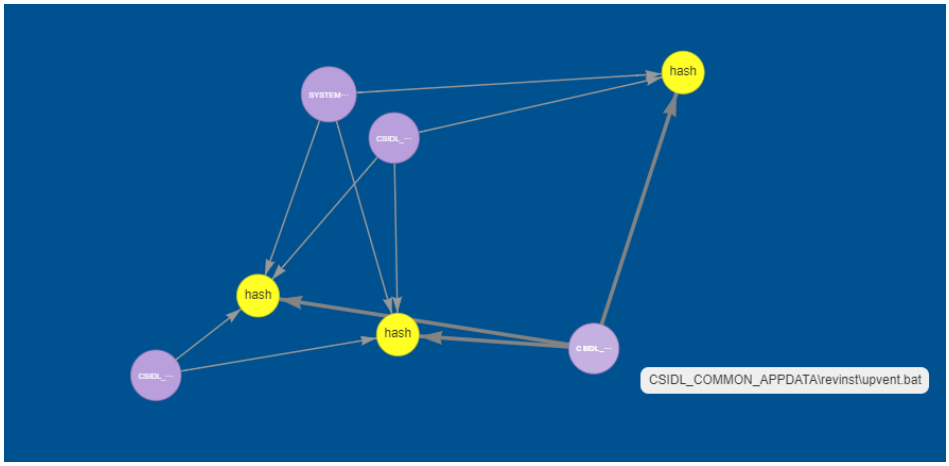
最终，Backdoor 和插件的大致功能可见下图。



三、关联样本分析

我们结合 VenusEye 威胁情报中心的历史数据关联到了与此次攻击样本相关

的更早的恶意代码样本。



下面我们针对关联到样本的同源特征进行分析。

3.1 配置信息比较

关联到的早期样本编译时间为 2014 至 2015 年(我们将这批样本分别称为 2014 类和 2015 类)。通过比较，发现在配置相关的字符串上，相关样本都保持了高度一致性。

main、main64、mainpath、commonappdata/temp0、64loadpath、loadpath、loadpath7、temp/upptent.exe、AfterInstallation、RemoveInstaller、twssrv.exe、KVSrvXP.exe

此前 Dropper 还有字符串 360tray.exe、commonappdata/**Revinst**/upvent.bat，本次的 Dropper 也用到了，但全是在栈里拼接的。

```
00403783                mov     [ebp+Str], '3'
00403789                mov     [ebp+var_36], '6'
0040378F                mov     [ebp+var_34], '0'
00403795                mov     [ebp+var_32], 't'
0040379B                mov     [ebp+var_30], 'r'
004037A1                mov     [ebp+var_2E], 'a'
004037A7                mov     [ebp+var_2C], 'y'
004037AD                mov     [ebp+var_2A], '.'
004037B3                mov     [ebp+var_28], 'e'
004037B9                mov     [ebp+var_26], 'x'
004037BF                mov     [ebp+var_24], 'e'
```

虽然跨了 3、4 年，字符串显然变化并不大，可预见的将来也不会大变。所以样本具有多个上述的字符串，因此可以归为同一个家族，我们按照字符串中一

个很独有的单词，将这个家族取名为 **Revinst**。

3.2 释放文件比较

除配置外，这三类的 Loader 即 temp0 也有共同字符串

series、mainpath、CommonAppData、System/、Windows/

而且 2014 类和 2015 类的 temp0 存在共同的字符串：

commonappdata/Windows CE/fake

这实际上是 fake 的路径。本次的 temp0 是加载图片资源来解密为 fake 路径。但 2014 和 2015 类的 temp0 没有对抗毒霸的模块 acess。

这三类的 fake 也有很多共有字符串：

loadpathxp、64loadpathxp、loadpath7、64loadpath7、loadpathsv、64loadpathsv、windows/explorer.exe、SearchIndexer-1、SearchIndexer-2、WorkMode、AddressList、ClientID 、 ControllerID 、 ControllerVersion 、 PluginCoder 、 Persistence 、 WorkingDirectory、system/msTracer.dll、windows/fxsst.dll、system/srvlic.dll、series 、 SpecialPlugin 、 SYSTEM\CurrentControlSet\services\SharedAccess\Parameters\FirewallPolicy\RestrictedServices\Static\System。

事实上，像 64loadpath7、system/msTracer.dll 等作为 Loader 的释放路径，fake 完全用不着，也确实没有用到，但在 fake 中仍然存在。如下图，很明显是配置和对应数据：

64loadpath7：system/msTracer.dll

64loadpathsv：windows/fxsst.dll

```
1000943C a64loadpath7:
1000943C                unicode 0, <64loadpath7>
10009452                dd 13h
10009456 aSystemMstracer:
10009456                unicode 0, <system/msTracer.dll>
1000947C                dd 0Ch
10009480 a64loadpathsv:
10009480                unicode 0, <64loadpathsv>
10009498                dd 11h
1000949C aWindowsFxsst_d:
1000949C                unicode 0, <windows/fxsst.dll>
```

攻击者写代码时，也许在各个模块里都直接复制拷贝了这些配置定义，不管是否用得到。在可预见的将来，字符串可能会有少许变化，但大多数会保持不变。

本次所使用的 fake 比 2014 和 2015 类的 fake 多了两个有关下载的 cfg 命令。命令码分别是 0C558B012/5047A6F4、0DDD7303E/0CDF2E7F4。

而三者的其它 4 个命令功能完全一样。

并且，三者的 fake 以及本次的 main 都有 ClientID、ControllerID 的配置。fake 的 ClientID 没发现对应的数据。在接收到 ClientBasicInformation 命令时，会回传 ControllerID 给 C&C。

	2014(fake)	2015(fake)	2017(本次 fake)	本次 main
ClientID	无	无	无	E5201314
ControllerID	20140410144726	20140410144726	201602241528288	201306281733338

ClientID、ControllerID 可能是攻击者用来标志区别样本的，也许某些样本只针对某些目标，当然只是猜测。

三类的 fake 和本次的 main 在和 C&C 通信时，通信格式、加密算法完全一致，其中的 Magic 也都是 0xC7315A6B。如果检测到可疑报文，解密后发现包含 0xC7315A6B，可以确认是这类木马。

显然总的看，在长达三四年的时间里，虽然代码有稍微变化，但大多数字符串或者命令，配置等都保持了一致。

四、溯源分析

通过对我们发现的早期样本进行溯源追踪后，发现相关样本与 GDATA 公司曾经披露的定向攻击行动“TooHash”中涉及的木马协议特征几乎一致。

以下为 fake 中的 opcode（见红框处）与历史披露的“TooHash”行动中的命令特征。

```
1  bool *__cdecl sub_1000311C(int a1, int a2, int Src, int a4)
2  {
3    if ( a1 == 0x134A6D30 && a2 == 0x100B4627 )
4      return sub_10002D44(Src, a4);
5    if ( a1 == 0xC558B012 && a2 == 0x5047A6F4 )
6      return sub_10002E61(Src, a4);
7    if ( a1 == 0x22836D73 && a2 == 0x6F42E3C0 )
8      return sub_1000251B(Src, a4);
9    if ( a1 == 0x3254BFD2 && a2 == 0x6FF39717 )
10     return sub_10002566(Src, a4);
11   if ( a1 == 0xB4749FFF && a2 == 0xA7109782 )
12     return sub_10002ECC(Src, a4);
13   if ( a1 == 0xDDD7303E && a2 == 0xCDF2E7F4 )
14     return sub_10003109(Src, a4);
15   return 0;
```

相关披露报告部分截图如下：

- If opcode1 == 0x3254BFD2 and opcode2 == 0x6FF39717
  → ExecCmd_LoadLibrary
- Command

SIZE[4]	SECRET_KEY[4]	MAGIC[4]	0x3254BFD2	0x6FF39717
NAME_LEN[4]	NAME[NAME_LEN*2]			

- Response

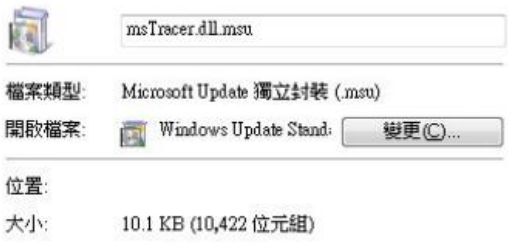
SIZE[4]	SECRET_KEY[4]	MAGIC[4]
MESSAGE_LEN[4]	MESSAGE[MESSAGE_LEN*2]	RETCODE[1]

用于绕过 UAC 的方法也完全一致。

启明星辰公司 版权所有

- makecab.exe /V1 "C:\Users\<USERNAME>\AppData\Local\Temp\msTracer.dll" "C:\Users\<USERNAME>\AppData\Local\Temp\msTracer.dll.msu"

- wusa.exe /quiet "C:\Users\<USERNAME>\AppData\Local\Temp\\msTracer.dll.msu" /extract:C:\Windows\system32



“TooHash”定向攻击行动疑似主要针对我国台湾地区进行攻击。其曾经使用过的攻击文件如下：

102 年尾牙、103 年春酒精緻菜單.xls

李辉简历.doc

文件列表.xls

联络表.xls

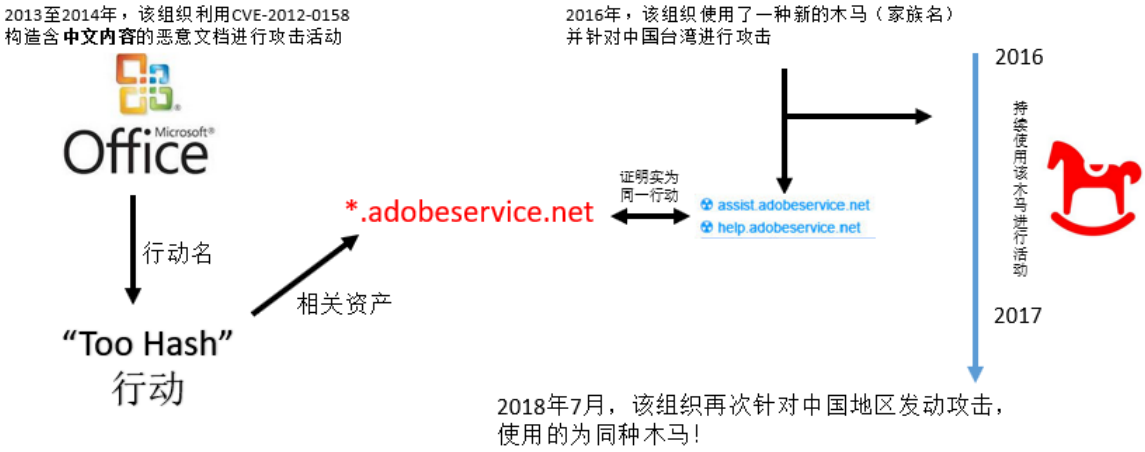
Wo.doc

结合最新发现的攻击样本、VenusEye 威胁情报中心关联到的历史样本以及友商披露的“TooHash”行动来看，都表明这是一个行事作风一致的定向攻击组织所为。并且最新攻击事件中的恶意程序相比之前无论是代码量，还是释放的组件都了许多，技术手段也复杂了许多，种种迹象表明该组织技术近几年一直在进步。

鉴于该组织是启明星辰独立发现并披露的，符合启明星辰金睛安全研究团队对 APT 组织进行独立命名的条件。故，我们将其命名为“狼毒草”。

同时，我们根据已经掌握的资料绘制出了该组织近几年的攻击时序图，可以看出其具有至少 5 年以上的活动期，并具有很强的隐匿性和杀伤力。

启明星辰公司 版权所有



从该组织投递的恶意软件中含有的大量检测与绕过中国杀毒软件的手段，可以确定攻击者深入研究了中国的安全防御体系，并将其转换为攻击力量。同时，结合攻击者熟练的中文水平以及高超的社工手段，对于我国必然是一个严重的威胁。

五、总结

“狼毒草”组织至少于 2013 年左右开始活动，期间偶有沉寂，如今再次伺机行动。种种迹象表明，“狼毒草”组织从未停止自己的活动，其使用的高级逃逸技术在不断进步，我们判断下一次攻击将会有更强的杀伤力，不容小觑。

目前，基于启明星辰 VenusEye 威胁情报中心的全线产品都已经支持对此次 APT 攻击活动的检测。

六、IOC

Hash：

ae3b268b4c48c385a788cdbfbb52ce0d

4dbd68d3741d46170d2585aae4336b80

a53a725a27eace541e52fc095e7af699

5cc7038f67114bfb64e0c23223d902b1

09a9bf5e8de0d9cfe3684b6df98b6d0d

启明星辰公司 版权所有

10d678deeaad0b45dea9bd70e21325da

a26ebe515cc6af6d05ad6d88c0257787

415027680047ed31fa5a84c94a46d582

31b6dbffc5f6d10e1fe7437626a7582b

40e916f03bbbc64921496e88d2d1d099

29daa2cffaf4e288388eb97da1f29a91

dd00fc9adaa3e20630dad5e5faaeeff5

66a4bd97867c6364a3846654dfd37a24

095cf3f0ef7111d386bf7312186c7656

1366f1c75368964f8af247d2709e4889

6ec1db9872f6ca979649b4dab7bbe7fc

024883786ba706fe8c8a6354a355d30c

PDB 路径：

main

Z:\z_code\Q1\Client\Win32\Release\MainPlugin.pdb

FxCoder

Z:\z_code\Q1\Client\Win32\Release\StreamCoderPlugin.pdb

inter

Z:\z_code\Q1\Client\Win32\Release\InterMainPlugin.pdb

Z:\z_code\Q1\Client\Win32\Release\InterMainStub.pdb

io

Z:\z_code\Q1\Client\Win32\Release\DebugPlugin.pdb

Z:\z_code\Q1\Client\Win32\Release\DebugStub.pdb

Z:\z_code\Q1\Client\Win32\Release\DebugDumper.pdb

utility

Z:\z_code\Q1\Client\Win32\Release\UtilityPlugin.pdb

启明星辰公司 版权所有

七、参考

1．https://hitcon.org/2016/pacific/0composition/pdf/1202/1202%20R0%200930%20an%20intelligance-driven%20approach%20to%20cyber%20defense.pdf

2．https://public.gdatasoftware.com/Presse/Publikationen/Whitepaper/EN/GDATA_TooHash_CaseStudy_102014_EN_v1.pdf

启明星辰公司 版权所有